ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP



Heft 1 · 1989

Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232-2892

KomCom '89:

MAP

Experimentalsystem für Mini-MAP

Transputer

- Innovative Architektur
- Einsatz in Spracherkennung und Robotik

Digitale Signalprozessoren

Signalanalyse und Echtzeitverarbeitung

Künstliche Intelligenz

1

2

3

4



5

1 Super 32

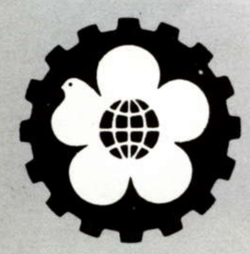
2 Prawez-386

3 CAD-System TIGS-R

4 Sport 386 und Sport 286 Quatro

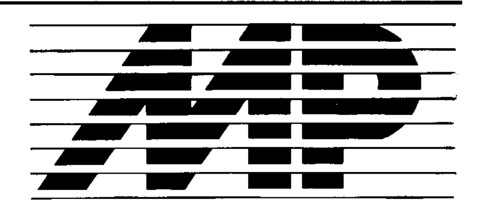
5 Bildverarbeitungssystem CSI-11

Lesen Sie dazu unseren Bericht am Schluß des Heftes



44. Internationale Technische Messe Plowdiw '88

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232-2892



Mikroprozessortechnik, Heft 1 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Herbert Hemke, Redakteur (Tel. 2870203); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelgrafik H. J. Eggstein

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß: 11. November 1988

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141-167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



Eine Einführung in das Gebiet der Künstlichen Intelligenz (KI) gibt Ihnen der Beitrag auf der **Seite 3**. Er betrachtet historische Wurzeln, Spezialgebiete und Tendenzen.
Mit diesem Artikel beginnen wir eine lose Folge von Beiträgen zum Thema KI.

KOMCOM '89

Mit vier Beiträgen, die auf der Fachtagung Kommunikations- und Computertechnik '89 zwischen dem 14. und 16. Februar 1989 in Dresden gehalten werden, machen wir Sie vorab in gekürzter Fassung bekannt:
Der Artikel „Experimentalsystem für Mini-MAP" auf der **Seite 7** gibt einen Überblick über internationale Entwicklungen zur schrittweisen Automatisierung in der Produktion.
Auf der **Seite 11** stellt der Beitrag „Innovative Computerarchitektur – Transputer" die gegenwärtig angebotenen Schaltkreise der Transputer-Familie, die Hochsprache OCCAM und potentiell vorhandene Anwendungsmöglichkeiten des Transputers in der Telekommunikation vor.
Einen zweiten Beitrag zum Thema Transputer finden Sie auf der **Seite 13**. Er befaßt sich mit dem diesen Prozessoren zugrunde liegenden Konzept, gibt eine Leistungseinschätzung der Transputer und stellt Applikationen für Spracherkennung und Robotik vor.
„Signalanalyse und Echtzeitverarbeitung mit digitalen Signalprozessoren" ist das Thema des Artikels auf der **Seite 19**. Vorgestellt wird die Realisierung der Autoleistungs-Spektralanalyse mit dem Signalverarbeitungssystem TITAN.

Vorschau
In MP 2/1989 finden Sie Beiträge zu folgenden Themen:
- Diskettenformate unter CP/M
- Anwendung des FDC U 8272 D
- 8 × 8 – ein Font für alle Fälle

Inhalt

Mehr automatisch bestückbare Leiterplatten aus Neuruppin

An einem automatischen Fertigungsabschnitt zur Leiterplattenherstellung in den Elektro-physikalischen Werken Neuruppin hat die Jugendbrigade „Manfred von Ardenne" nach halbjähriger Einlaufphase im September die volle projektierte Leistung erreicht. Damit ist der Betrieb in der Lage, den stark gestiegenen Bedarf an automatisch bestückbaren Leiterplatten in der DDR zu decken.
Rund 35000 Quadratmeter unterschiedlicher Leiterplatten liefert das Neuruppiner Werk mit seinen 3300 Beschäftigten monatlich an die Elektonikindustrie der DDR und ist damit wichtigster Hersteller. *ADN*

Gemeinsame Forschung für Chipproduktion

Mit durchschnittlich 80 bis 100 neuen Typen erweitert das Kombinat Mikroelektronik Erfurt jährlich sein Produktionsprogramm elektronischer Bauelemente. Vorlauf für neue Schaltkreisgenerationen, wie schnelle Mikroprozessorsysteme mit hohem Integrationsgrad, wird in Zusammenarbeit mit über 30 Hoch- und Fachschulen sowie Instituten der Akademie der Wissenschaften der DDR geschaffen. Grundlage für die Forschungskooperation sind langfristige Koordinierungs- und Leistungsverträge. *ADN*

Neues Werk für Leiterplatten in Bulgarien

Ein Werk für Leiterplatten ist jetzt in Prawez, 70 Kilometer nordöstlich von Sofia, eingeweiht worden. Der Betrieb, der zum Kombinat für Mikroprozessortechnik Prawez gehört, wurde innerhalb kurzer Zeit errichtet: Im September 1985 war der Beschluß gefaßt worden, entsprechend der Orientierung der BKP auf beschleunigte Entwicklung der Mikroelektronik ein derartiges Werk zu bauen, und bereits am 21. Dezember 1987 verließ die erste Leiterplatte die Produktionsabteilung.
In seiner ersten Ausbaustufe sichert das Werk den Bedarf des Kombinats an Leiterplatten. Es soll bis Ende 1990 seine volle Kapazität erreicht haben und ausschließlich mit CAD/CAM-Technik projektieren und produzieren. *ADN*

USA-Elektronikindustrie will in die Offensive

Die USA-Elektronikindustrie will auf den Außenmärkten gemeinsam in die Offensive gehen, um dem Ansturm japanischer und westeuropäischer Konkurrenz zu begegnen. Vor wenigen Wochen trafen sich im Washingtoner Capitol Manager der Branche mit Kongreßabgeordneten und Wissenschaftlern zu einer Art Strategieklausur. Eingeladen hatte „Rebuild America", eine vom Mitglied des Repräsentantenhauses Mel Levine aus dem Silicon Valley geleitete Gruppe. „Wir sind dabei, zum ersten mal im 20. Jahrhundert unsere Stellung als Spitzenreiter in der Hochtechnologie zu verlieren", warnte Levine. Er fand bei den Männern aus der Industrie besorgte Zustimmung.
John Roach, Chef der Tandy Corporation, Richard Elkus von Prometrix, Jack Tramiel, Begründer der Atari Corporation, und andere nannten Beispiele dafür, wie USA-Entwicklungen auf dem Gebiet der Videotechnik, der Personalcomputer und weiterer Bereiche der Heimelektronik kein kommerzieller Erfolg wurden. Es habe wenig Sinn, über die ausländische Konkurrenz zu klagen, meinte Mel Levine. Die meisten Probleme seien hausgemacht.
Soweit Vorstellungen der Branche an die Öffentlichkeit drangen, läßt sich folgendes Herangehen erkennen: Nach japanischem Vorbild wird eine nationale Industriestrategie angestrebt, die von einer „neuen politischen Führung" mitgetragen werden müsse und alle entscheidenden Positionen von künftigen Speichern über die Lichtleitertechnik und die nächste Generation von Fernsehgeräten bis zu Großrechnern erfassen soll. Das Weltraumrüstungsprogramm SDI gilt nicht als Lösung. Von der Regierung wird eine Umverteilung von Steuermitteln zugunsten ziviler Forschung erwartet.
Jack Tramiel von Atari will sogar die Schaffung eines Export-Ministeriums und einen Marshallplan für die Elektronikindustrie. Bemerkenswert ist jedoch vor allem, daß Spitzenleute eines der entscheidenden Industriezweige des Landes mit Ambitionen außerhalb des militärisch-industriellen Komplexes unmißverständlich ihre Wünsche an die Regierung nach Reagan formulierten.
ADN

Hinweise für das Anfertigen von Manuskripten

1. Die mit Maschine einseitig beschriebenen Manuskripte – 2zeilig, 60 Anschläge/Zeile, 30 Zeilen/Seite – senden Sie bitte in zweifacher Ausfertigung ein.
Als Manuskript können auch Computerausdrucke mit einem sauberen und kontrastreichen Schriftbild dienen, sofern durchschnittlich 60 Anschläge je Zeile eingehalten werden (also wie bei Schreibmaschine **mit** Silbentrennung, kein Randausgleich!). Bei Verwendung von Matrixdruckern ist mindestens Near Letter Quality (NLQ) notwendig.
2. Der Umfang des Manuskriptes soll im allgemeinen 10 Seiten nicht überschreiten. Die maximale Länge des Textteiles von Angeboten und Suchmeldungen für die Rubrik „Börse" beträgt 30 Zeilen.
3. Das Manuskript beginnt mit einer aussagekräftigen, aber knappen Überschrift; es folgen der volle Vor- und Zuname, der akademische Grad und bei Beiträgen aus ihrem Arbeitsbereich die Institution.
4. Versuchen Sie, dem Leser die Aufnahme der Informationen zu erleichtern, indem Sie bei der Gestaltung des Textteiles folgendes beachten:
- Beitrag übersichtlich gliedern! (Zwischenüberschriften, Absätze; evtl. kurze Zusammenfassungen, Hervorhebung von Kernsätzen)
- Wo es der Aussage oder Anschaulichkeit dient, verwenden Sie bitte Bilder, Tafeln und Fotos (s. Pkt. 6).

5. Werden Abkürzungen verwendet, sind die im Duden aufgeführten zu nutzen; zusätzlich abzukürzende Begriffe sind beim ersten Auftreten auszuschreiben, und die Abkürzung ist in Klammern zu setzen.
6. Bilder, Tafeln und Fotos müssen als Anhang getrennt numeriert beigelegt werden. Die Einordnung ist im Text zu kennzeichnen. Für die erläuternden Unterschriften – sie sind bei allen Bildern, Tafeln und Fotos notwendig! – ist ein gesondertes Manuskriptblatt zu verwenden.
Bilder und Tafeln können als übersichtliche und eindeutig lesbare Bleistiftzeichnung bzw. maschinengeschrieben eingereicht werden. Um Satzfehler, insbesondere bei Listings zu vermeiden, sind diese **einzeilig** als reprofähige Vorlage zu liefern (kontrastreicher Druck auf weißem Papier, auch saubere Schreibmaschinenschrift ist möglich).
Schwarzweißfotos sollten etwa das Format 13 cm × 18 cm haben (das Negativ wird nicht benötigt).
Farbfotos stimmen Sie bitte mit der Redaktion ab. Geben Sie bei honorarpflichtigen Fotos den Namen und die Adresse bzw. Konto-Nr. des Bildautors an!
7. Bedenken Sie, daß die Anzahl der Zeichen pro Zeile im Druck geringer ist als im Manuskript. Passen Sie also im Interesse einer korrekten Wiedergabe Gleichungen, Formeln, kurze Programmauszüge u. ä., die im laufenden Text erscheinen sollen, von vornherein an die **Spaltenbreite** (etwa 40 Anschläge) an.
Wo dieses nicht möglich ist, sind diese Zeilen als Bilder zu deklarieren.
8. Literaturverzeichnisse werden, in Schrägstriche eingeschlossen, fortlaufend numeriert, z. B. „wie in /1/ und /2/ eindeutig dargestellt wurde", oder „auch Meyer hat in /3/ darauf hingewiesen". Das Verzeichnis der verwendeten Literatur wird auf einem gesonderten Manuskriptblatt nach folgendem Schema angefertigt:

Literatur
/1/ Menzer, R.; Richter, B.: Neue Technologien für Digitalisiergeräte. Feingerätetechnik, Berlin 34 (1985) 9, S. 386
/2/ Claßen, L.; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. VEB Verlag Technik, Berlin 1986

9. Jedem Fachartikel ist gesondert ein etwa 10 Zeilen umfassender **Referateteil** beizufügen, der das Wesentliche des Beitrages beinhaltet.
10. Bei Fachaufsätzen ist ein kurzes **Autorporträt** – mit Angaben über Alter, Ausbildung, beruflichen Werdegang, die jetzige Tätigkeit und Aufgabenschwerpunkte – erwünscht.
11. Unterstützen Sie bitte unsere Bemühungen, den Erfahrungsaustausch zu fördern, indem Sie für Interessenten eine Kontaktadresse und Telefonnummer angeben.
12. Denken Sie daran, bei Fachbeiträgen aus Ihrem Arbeitsbereich die Veröffentlichungsgenehmigung der Dienststelle beizulegen.
13. Nicht zu vergessen sind schließlich die private und dienstliche Anschrift, Ihre Konto-Nr. sowie für evtl. Rückfragen eine Telefonnummer.

Bitte überprüfen Sie vor dem Absenden des Manuskriptes noch einmal die Einhaltung dieser Hinweise – Sie vermeiden damit unnötige Verzögerungen bei der Bearbeitung und Veröffentlichung Ihres Beitrages.

Ihre Redaktion MP

Schritte zum Neurocomputer

In der BRD begannen unlängst neun Forschungsgruppen mit einem Programm, das zur Neuroinformatik führen soll. Ziel ist es, neue Wege zur Entwicklung von Computern zu finden, die nach dem Prinzip menschlicher Gehirnstrukturen funktionieren. Die daran beteiligten Forscher wollen moderne Erkenntnisse über die prinzipielle Arbeitsweise des Gehirns in Verfahren der Informationsverarbeitung umsetzen.
Als erste Forschungsthemen wurden Aufgaben formuliert, die im Gehirn scheinbar mühelos, von heutigen Computern dagegen nur unvollkommen gelöst werden können: die Verarbeitung von Bildfolgen aus der natürlichen Umgebung und die Bewegungssteuerung für autonome Roboter.
Die Beteiligten an dem Projekt – neben der Max-Planck-Gesellschaft und einer Industriefirma verschiedene Universitäten – gehen davon aus, daß die Entwicklung eines universellen, neuronalen Computers erst als Fernziel zu verwirklichen ist. Deshalb werden sie in den ersten Jahren die technischen Nutzungsmöglichkeiten neuronaler Prinzipien, wie die Organisation von Wissen oder die Eigenschaften paralleler Netzwerkarchitekturen, mit einfachen Funktionsmodellen ausloten.
aus ND vom 15./16. Oktober 1988

Künstliche Intelligenz in Gegenwart und Zukunft

Prof. Dr. Christian Posthoff, Dr. Rainer Staudte
Technische Universität Karl-Marx-Stadt, Sektion Informatik

Fragen aus dem Gebiet der Künstlichen Intelligenz (KI) rückten in letzter Zeit wieder mehr in den Mittelpunkt der Aufmerksamkeit. Diese Rennaissance hat ihre objektiven Ursachen in der Notwendigkeit, alle produktiven und nichtproduktiven Bereiche der Volkswirtschaft durch die gezielte Nutzung modernster Technologien effektiver zu gestalten und auf eine qualitativ höhere Stufe zu heben. Ziel dieses Beitrages ist eine kurze Einführung in den Problemkreis. Diese Thematik wird in der Folge durch spezielle Themen weitergeführt. Nicht diskutiert werden soll der Begriff Künstliche Intelligenz selbst, der sicher nicht glücklich ausgewählt wurde, aus historischen Gründen aber heute nicht mehr ausgemerzt werden kann.

Im Laufe ihrer Entwicklung hat sich die KI zu einer interdisziplinären Verflechtung von Informatik, Mathematik, Psychologie und ihren Anwendungsgebieten herausgebildet. Einige Aufgaben, die früher zweifellos zum Problemkreis der KI gerechnet wurden und später weitgehend einer Lösung zugeführt werden konnten, zählen heutzutage zum Allgemeinwissen der Informatik. Andere Aufgaben, deren Lösung sich prinzipiell als schwieriger erwiesen hat, brachten selbständige Teildisziplinen hervor. Für die Zukunft ist das Verschwinden des Begriffes durchaus denkbar, weil er eines Tages inhaltlich nicht mehr existieren wird und die Mittel und Methoden zur Lösung der Aufgaben dieser Klasse eine vernünftigere Einteilung bestimmt haben werden.

Die Wurzeln der KI

Geht man von den Aufgaben aus, die unter dem Begriff KI vereinigt sind, so wird sehr schnell die Notwendigkeit deutlich, ein breites Spektrum wissenschaftlicher Forschung zu verfolgen. Mindestens drei Entwicklungslinien des 19. bzw. 20. Jahrhunderts, die ihrerseits noch weiter in die Geschichte zurückreichen, sind für den heutigen Stand der KI von grundlegender Bedeutung:

① die Mathematische Logik, die das logische Folgern als ein Teilgebiet menschlichen Denkens als formalisierbar nachwies, die Aussagen- und Prädikatenlogik hervorbrachte und sich gegenwärtig intensiv mit der Informatik im allgemeinen und der KI im besonderen vermischt. Das Verhältnis *Mathematische Logik – Informatik* wird sich sicher ebenso unauflöslich eng und fruchtbar gestalten wie die Beziehung *Mathematik – Physik*. Namen wie *Boole, Frege, Whitehead, Russell, Hilbert, Tarski* und *Gödel* kennzeichnen diese Linie.

② ein zunehmendes Verständnis des Berechenbarkeits- und Algorithmusbegriffs, ein theoretisches Bindeglied zu den zum damaligen Zeitpunkt noch nicht existierenden Computern und Computersystemen. Die „abstrakten Maschinen" von *Turing* und *Post* als Denkmodelle zur Charakterisierung der Berechenbarkeit, die Theorie der rekursiven Funktionen, die *Church*sche These als Verbindung zwischen diesen Denkmodellen in intuitiven Berechenbarkeitsvorstellungen und nicht zuletzt bereits hier die grundlegende Vorstellung, mit derartigen Maschinen *Symbole* (nicht etwa nur Zahlen) zu verarbeiten – das sind die Ergebnisse, die einen wichtigen exakten Ausgangspunkt für die Entwicklung intelligenter Computersysteme bilden.

③ die eigentliche konstruktiv-technische Entwicklung und Realisierung von Computersystemen selbst, die Mitte der vierziger Jahre begann und sich bis in die heutige Zeit (und wohl auch noch in Zukunft) mit ungeheuerer Geschwindigkeit und Breite fortsetzt. Namen wie *Leibniz, Babbage* (als Vorläufer im 19. Jahrhundert bereits alle wichtigen Strukturelemente eines Computers nennend), *Turing, von Neumann, Zuse* stehen als Beispiele. Die fünfte Computergeneration, verknüpft mit dem Namen *Moto-Oka,* ist ein reales Ziel. Und bis auf *Babbage* und seine Schülerin *Ada Augusta Countess of Lovelace* geht auch die Idee zurück, in derartigen Komplexen (mechanischen, später elektronischen Systemen) intelligentes Verhalten zu realisieren. Er traut seiner „Analytical Engine" die Fähigkeit des Schachspielens zu, eine Idee, die, 1948 von *Shannon* erneut ausgesprochen und prinzipiell entwikkelt, bis heute intensiv weiterverfolgt wird.

Zum Gegenstand der KI

Vielleicht ist es gar nicht so absurd, nach Erscheinen der ersten Computer und erst recht heute bei der vorhandenen Massenbasis an Computersystemen, das Bestreben, den Computer bis an die Grenze seiner Leistungsfähigkeit auszureizen, als eine wichtige Triebkraft für die Entwicklung der KI anzusehen.

Verfolgt man die aktuellen KI-Entwicklungen, so findet man nach wie vor ein recht heterogenes Bild vor, das durch einen *anthropomorphen* Standpunkt systematisiert wird und eine äußere, problembezogene Hülle darstellt. Nachfolgender, von *Siekmann* ausgesprochener Standpunkt soll als Arbeitsgrundlage dienen:

„Gewisse menschliche Aktivitäten wie das Planen einer kombinierten Bahn-Bus-Reise, das Verstehen gesprochener Sprache, das Beweisen mathematischer Sätze, das Erstellen einer medizinischen Diagnose oder das Sehen und Erkennen bestimmter Gegenstände erfordern zweifellos Intelligenz – unabhängig davon, welche Definition dieses Begriffes man bevorzugt. Die *Künstliche Intelligenz* faßt diese bisher dem Menschen vorbehaltenen kognitiven (auf Erkenntnis beruhenden) Fähigkeiten als informationsverarbeitende Prozesse auf und macht sie naturwissenschaftlichen Untersuchungsmethoden (*und* ingenieurmäßiger Verwendung) zugänglich."

Die Unterstreichungen sollen den anthropomorphen Standpunkt direkt sichtbar machen: die Fähigkeiten, abstrakt, antizipierend (gedanklich vorwegnehmend) zu planen, zu beweisen, über seine Sprache die Umwelt zu bewältigen, zu sehen – diese Möglichkeiten des Menschen stellen also Aufgaben für die KI dar. Einige menschliche Fähigkeiten sollen ganz oder teilweise irgendwie auf Computer verlagert werden, mindestens mit den Zielen:

- geht das überhaupt?
- kann man das für die Praxis (mit vertretbarem Aufwand) realisieren?

Spezialgebiete der KI

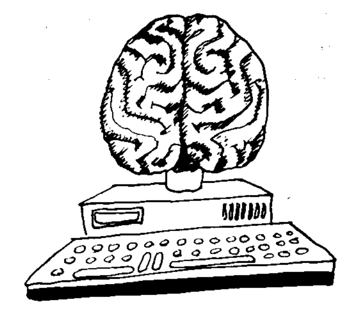
Auf die Vielzahl von philosophischen, moralischen, ethischen und sozialen Problemen, die sich hinter diesen lakonischen Fragen verbergen, sei hier noch einmal nachdrücklich hingewiesen.

Somit lassen sich Spezialgebiete der KI formulieren, wobei aber grundlegend zu betonen ist, daß Computersysteme menschliche Mittel und Möglichkeiten in *allen* Bereichen

Kleines Lexikon der KI

Künstliche Intelligenz

Unter dieser Überschrift formiert sich in den letzten 30 Jahren ein Wissenschaftsgebiet, das im wesentlichen als Teilgebiet der *Informatik* angesehen werden kann, in seinen theoretischen Grundlagen starke Beziehungen zur *Mathematik* aufweist (Diskrete Mathematik, Mathematische Logik, Graphsuchverfahren, ...), zunehmend mit den Geisteswissenschaften in Beziehung tritt (*Linguistik, Philosophie, Psychologie*) und versucht, Probleme auf Computersystemen lösbar zu machen, deren Bewältigung sich auf Informationsverarbeitung reduzieren läßt und beim Menschen Intelligenz erfordert (wie auch immer *„Information"* und *„Intelligenz"* erklärt oder definiert werden sollen).



Damit gelangt man zu vielen verschiedenen „Definitionen" der KI, die im wesentlichen immer die Intentionen der Autoren ausdrücken. Nach wie vor handelt es sich um ein sehr heterogenes Gebiet, in dem eine anthropomorphe Klammer oft der einzige Zusammenhang ist (das heißt der Wunsch, menschliche Fähigkeiten nachzubilden). Sehr intensiv und bedeutsam ist vor allem die Rückwirkung auf die Informatik selbst (Sprachen der KI, Hardwarestrukturen der KI, Automatisches Programmieren u. v. a. m.).

Kleines Lexikon der KI

Expertensysteme



Expertensysteme sind der anwendungsbezogene Aspekt der Wissensverarbeitung. Es handelt sich um Programmsysteme, die einen Menschen in einem bestimmten Problembereich ersetzen oder unterstützen sollen, in dem (noch) keine exakten Lösungen existieren. Derartige Gebiete werden häufig von Menschen bewältigt, die viel „Erfahrung" besitzen (eben Experten sind), die kaum zu einer präzisen Arbeitsmethode umgewandelt werden kann und die im wesentlichen an den Experten gebunden ist und von diesem im allgemeinen (falls überhaupt) durch Ausbildung (Belehrung) jüngerer Experten weitergegeben wird.
Typische Forschungs- und erste Anwendungsbereiche sind beispielsweise Diagnosesysteme im technischen und medizinischen Bereich, Planungssysteme, Beratungs- und Auskunftssysteme, Dispatcher-, Steuer- und Entscheidungssysteme usw. usf. Die meisten derartigen Systeme tragen gegenwärtig experimentellen Charakter; bilden jedoch einen wichtigen Ausgangspunkt für eine gewaltige Entwicklung aller Bereiche der menschlichen intelligenten Tätigkeit. Expertensysteme besitzen eine *Wissensbasis*, eine Problemlösungskomponente (*Inferenzmaschine*), können mit dem Nutzer kommunizieren (*Dialogkomponente*), und (eventuell) ihre Lösung erklären (*Erklärungskomponente*). Damit wird der Übergang zu *intelligenten Lehrsystemen* fließend.

Wissensverarbeitung

Die Wissensverarbeitung mit den Vorstufen der *Wissensgewinnung* und der *Wissensdarstellung* bildet gegenwärtig ein Kerngebiet der KI. Man benötigt zur Lösung einer Aufgabe (eines Problems) eine gewisse Menge an **Wissen** (soviel wie nötig, sowenig wie möglich), wobei das erste zu lösende Problem (oft das schwierigste) darin besteht, einem Experten (falls vorhanden) dieses Wissen „abzuringen", meist ein langwieriger, iterativer Prozeß, der häufig in der jeweiligen Disziplin viele neue Fragen und Probleme aufwirft.
Für die Darstellung von Wissen (in Computersystemen) gibt es bereits eine Reihe gut ausgearbeiteter und beherrschter Formalismen, mit denen man sich in etwa vertraut machen muß, weil eine effektive Problemlösung sehr häufig von einer günstigen (geschickten, ...) Darstellung abhängt. Dabei sind die unterschiedlichsten Gesichtspunkte zu beachten: Sicherheit des Wissens (präzise, unsicher, zweifelhaft, ...), Verwendungszweck (Definition, Klassifikation), Beschaffenheit (Struktur, Regelwerk, ...), Metawissen. Häufig verwendet man einen objekt- und rational orientierten Standpunkt: Man definiert Objekte, charakterisiert diese durch Eigenschaften und beschreibt die zwischen ihnen vorhandenen Relationen (kausal, temporal, ...). Wichtige Repräsentationsformalismen sind:
- prozedurale Repräsentationen (endliche Automaten, Programme)
- Aussagen- und Prädikatenlogik, unscharfe und weitere nichtklassische Logiken
- Semantische Netze
- Regeln
- Frames
- objektorientierte Sprachen (Weiterentwicklungen von Simula und Smalltalk).

Zu jeder Darstellung gehört die entsprechende Inferenzmaschine, mit deren Hilfe die dann zu lösenden Probleme bewältigt werden.
Zunehmend treten diese Darstellungen gemischt und kontinuiert auf und werden durch geeignete Softwarewerkzeuge und Entwicklungsumgebungen unterstützt.

Lisp

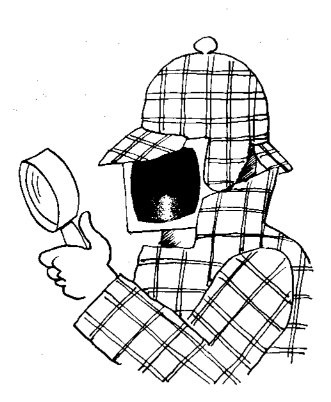
Lisp ist eine der ältesten Programmiersprachen (nur Fortran ist älter) und bis heute *die* Sprache der KI. Sie wurde Ende der 50er von J. McCarthy erfunden und steht der mathematischen Logik (dem Lambda-Kalbu) sehr nahe. Dennoch ist es eine vollwertige Programmiersprache und wurde als Sprache zur Verarbeitung von Listen (List processing) entworfen, um günstige Möglichkeiten zur Behandlung von Symbolverarbeitungsproblemen zu haben (symbolische Differentiation und Integration von algebraischen Ausdrükken). Im Laufe der Zeit hat sich Lisp auf Grund der Eleganz und Flexibilität seiner Konzeption an die modernsten Ideen des Software-Entwurfes angepaßt und wird heute durch eine ganze Sprachfamilie repräsentiert. Lisp gehört zu den funktionalen Sprachen und hat die bedeutsame Eigenschaft, daß Lisp-Programme selbst als Lisp-Daten angesehen und behandelt werden können. Damit ist Lisp auch ein elegantes Mittel, um Interpreter und Compiler für andere Sprachen zu schreiben.

Prolog

Prolog bildet heute (in unterschiedlichsten Realisierungen) die bekannteste Entwicklung aus dem Bereich der Logischen Programmierung. Es entstand ab 1972 in Frankreich (A. Colmerauer) und beruht auf dem Automatischen Theorembeweisen, einer Realisierung von Beweisverfahren der Prädikatenlogik auf Computersystemen.
Mit Hilfe von *Fakten* und *Regeln* wird ein Problemraum definiert, so daß ein logisches Schließen über Eigenschaften dieses Raumes möglich wird. Dies wird immer dann nötig, wenn eine Anfrage an das System gerichtet wird. Das System beweist das Vorhandensein oder Fehlen dieser Eigenschaften, wobei auftretende Bedingungen als Werte von entsprechenden Variablen ausgedrückt werden. Prologsysteme stellen ein reichhaltiges Repertoire vom System realisierter Aufgaben zur Verfügung (Rekursion, Backtracking, eingebaute Prädikate), so daß ein Programmierer sich primär auf die Beschreibung des Problemraumes konzentrieren und eine hohe Ausdrucksfähigkeit erreichen kann (*deskriptiver Aspekt*). Bei der Abbildung auf eine konkrete Maschine entstehen darüber hinaus auch prozedurale Eigenschaften von Prologprogrammen (*prozeduraler Aspekt*). Die Anwendungsprogrammierung der nächsten Jahre wird in hohem Maße von Methoden der logischen Programmierung beeinflußt werden.

Deduktionssysteme

Hierunter versteht man Programmsysteme, die Teilgebiete des „logischen Denkens" automatisieren (siehe auch Prolog) und auf Kalkülen der Mathematischen Logik beruhen. Die Grundidee besteht darin, daß gewisse, nach bestimmten Vorschriften konstruierte Formeln, sogenannte *Axiome*, als wahr (gültig, richtig) angesehen werden; mit Hilfe bestimmter *Schlußregeln* kann man aus diesen Axiomen neue *Theoreme* (Sätze, Gesetze) einer bestimmten Theorie ableiten. Findet man in der objektiven Realität eine *Interpretation* (eine Zuordnung der verwendeten Formelzeichen zu konkreten Objekten, Relationen, Funktionen, ...), die die Axiome erfüllt, so gelten auch alle gefundenen Theoreme.



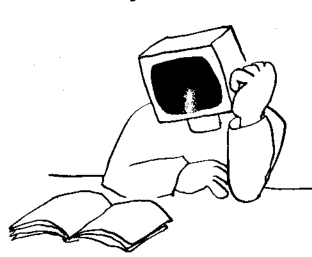
Deduktionssysteme hatten am Anfang hauptsächlich für Teilgebiete der Mathematik und der Mathematischen Logik Bedeutung, gehen aber heute schon weit darüber hinaus und gewinnen ständig an Bedeutung (logische Programmiersprachen, deduktive Datenbanken, Nachweis der Korrektheit von Hard- und Software, Verifikation von Mikroprogrammen, Konstruktion von Inferenzmaschinen, deduktive Planungsmethoden usw.).

Verarbeitung der natürlichen Sprache

Dieses Gebiet ist, von den Fähigkeiten des Menschen aus betrachtet, ein Schlüssel zur Nachbildung menschlicher Intelligenz, da ja die Sprache Ausdruck des menschlichen Denkens ist und demzufolge eine große Herausforderung an die KI darstellt.
Hier ist das Forschungsgebiet nahezu unüberschaubar und enthält eine Vielzahl schwierigster Probleme. Obwohl bereits beeindrukkende Erfolge und Ergebnisse auf Teilgebieten erreicht wurden, ist man von einer vollständigen Beherrschung der natürlichen Sprache noch weit entfernt.
Wichtige Teilgebiete und Forschungsaufgaben bestehen beispielsweise in der Schaffung von Systemen für:
- Dialoge zwischen Mensch und Maschine in natürlicher Sprache
- die automatische Wissensgewinnung aus Texten
- die automatische Übersetzung von einer Sprache in eine andere
- die akustische Ein- und Ausgabe (die Umsetzung des gesprochenen Wortes in Texte).

Ähnlich kompliziert gestaltet sich die Situation auch für die *automatische Bildverarbeitung*.

Lernende Systeme



Lernen und Wissen stehen natürlich ebenfalls in enger Beziehung zur menschlichen Intelligenz. Primitiv betrachtet, kann man Lernen stets als Modifikation vorhandenen Wissens ansehen mit dem Ziel, die Problemlösefähigkeit eines Menschen (und auch eines KI-Systems) zu erhöhen. Damit ist das Lernen auch eine Methode des Wissenserwerbs. Das Erlernen von Lösungsmethoden und -strategien, die Strukturierung von Wissen, die Entdeckung neuer Relationen, Fakten und Theorien, die Verbesserung heuristischer Vorgehensweisen, alles das in Wechselwirkung mit einem bestimmten Anwendungsbereich, gehört ebenfalls in dieses Gebiet.
Analoges Schließen, Verallgemeinerung und Induktion, Heuristiken, das Lernen auf Grund von Beispielen, die Gewinnung von Regeln aus Beispielen – alles das sind menschliche Fähigkeiten, die zur Zeit untersucht und modelliert werden; immer bewirkt die Beschäftigung mit der Künstlichen auch den Respekt vor der natürlichen Intelligenz.

Zeichnungen: Eggstein (4)

ergänzen, erweitern, beeinflussen werden. Ergebnisse dieser Spezialgebiete erweitern und vervollkommnen den Werkzeugsatz der Computeranwendungen, den es zu ordnen und zu systematisieren gilt.
Entsprechend unserem Ausgangspunkt lassen sich folgende Richtungen definieren:

- Problemlösen
- Verarbeitung natürlicher Sprache
- Bilderkennung und -verarbeitung
- Robotik
- Expertensysteme
- Kognition.

Auch diese Gebiete sind inhaltlich keineswegs voneinander abgegrenzt, sondern durchdringen, ergänzen, befruchten sich wechselseitig (sehende Roboter, bildverarbeitende Expertensysteme, ...). In einem folgenden Beitrag wird ausführlicher auf das Problemlösen mit den verschiedenen heuristischen Suchverfahren als Grundlage zur Entwicklung von Inferenzstrategien und auf Expertensysteme als besonders praxisrelevante Seite der KI-Entwicklungen näher eingegangen.

Tendenzen in der KI

Anfang der siebziger Jahre verlagerte sich der Schwerpunkt der Forschungsarbeiten der KI. Problemlösen und heuristische Suche verloren ihre dominierende Stellung. Dafür gab es zwei Ursachen: Zum einen war dieser Problemkreis bereits recht gut ausgearbeitet, andererseits zeigten die Erfahrungen, daß das Streben nach effektiven Suchmethoden im Zustandsraum für die Bewältigung einer Reihe komplexerer praxisrelevanter Aufgaben allein nicht ausreichte. Geeignete Wege, problemspezifisches *Wissen* in die Lösung der gestellten Aufgaben mit einzubeziehen, waren zu finden. In der Folgezeit rückten Fragen der Wissensdarstellung (Wissensrepräsentation), der Wissensverarbeitung (Wissensmanipulation) und des Wissenserwerbs (Wissensakquisition) in den Mittelpunkt. Mit der weiteren Entwicklung der Informatik, insbesondere der Softwaretechnologie, der Programmiersprachen mit den abstrakten Datentypen, der Datenbanken, der Rechnerarchitektur usw., konnten wesentliche theoretische und praktische Voraussetzungen zur Beherrschung informationeller Prozesse geschaffen werden. Hinter Wissensverarbeitung verbirgt sich heutzutage die Frage, wie riesige strukturierte Datenmengen geeignet ermittelt, bereitgestellt, verwaltet, umgeordnet, erweitert und verarbeitet werden können.
Im Laufe der Zeit bildeten sich – neben zahlreichen Fehlschlägen – fundierte Mittel und Methoden als Ergebnis dieser Ingenieurtätigkeit heraus: Expertensysteme, Datenbanksysteme, KI-Sprachen, Modelle der Wissensverarbeitung (Produktionssysteme, Frames, semantische Netze, ...). Induziert durch die stürmische Entwicklung der Mikroelektronik wirkte die Konstruktion ständig leistungsfähigerer Computer als Katalysator in dieser Entwicklung. Diese relativ rasche Entwicklung der Hardware machte sich nicht nur durch die Verbesserung wichtiger Parameter (Operationsgeschwindigkeit, Speicherplatz) bemerkbar, sondern brachte auch eine neue Qualität von Computern hervor. Diese Personalcomputer zeichnen sich durch ihre vergleichsweise niedrigen Herstellungskosten aus und gestatten ihre wirtschaftliche Nutzung in vielen neuen Anwendungsbereichen. Gegenwärtig sind die Methoden der heuristischen Suche und des formalen logischen Folgerns auf dem Computer bereits relativ weit entwickelt und bilden das Fundament jedes KI-Systems.
Die Eroberung nichttraditioneller Bereiche mit dem Computer, Hauptziel der KI seit ihrer Existenz, ist insbesondere für kleine Länder mit begrenzten ökonomischen Ressourcen, wie die DDR, unumgänglich geworden. Begonnen werden kann damit kaum, indem fertige Softwaresysteme einfach in einen neuen Problembereich umgesetzt werden, sondern nur über den evolutionären Weg des Verstehens und qualifizierten Anwendens der erwähnten Mittel und Methoden der Informatik. Das gilt auch für die in letzter Zeit aufkommenden Rahmensysteme (Shells), die das Begreifen der in ihnen ablaufenden Prozesse nicht überflüssig machen, sondern im Gegenteil dringend auf die Tagesordnung setzen.

☒ KONTAKT ☎
Technische Universität Karl-Marx-Stadt, Sektion Informatik, Straße der Nationen 62, Karl-Marx-Stadt, 9010; Tel. 668529

Programmieren in Prolog

Prozedurale Interpretation und Programmabarbeitung

Dr. Harald Killenberg, Rainer Knauf
Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik

Dieser Beitrag ist inhaltlich die Fortsetzung des in MP 11/87 erschienenen Artikels /1/, in dem eine Einführung in die logischen Grundlagen der Programmiersprache Prolog gegeben wurde.
Hier wird zunächst die prozedurale Sichtweise der deklarativen Sichtweise gegenübergestellt. Im Anschluß daran werden die in gewissem Umfang mögliche Steuerung der Abarbeitung von Prolog-Programmen behandelt, auf häufige Programmierfehler hingewiesen sowie ein Anwendungsbeispiel vorgestellt.

Prolog aus prozeduraler Sicht

Neben der (bisher betrachteten) deklarativen Interpretation gibt es noch eine prozedurale Interpretation von Prolog-Programmen. Dies soll an einem kleinen Beispiel verdeutlicht werden, in welchem von den im Abschnitt 5 /1/ definierten Prädikaten „chef_von" und „mitglied" Gebrauch gemacht wird:

(1) weisungsberechtigt (K1,K2):–
chef_von (Kollegen,K2),
mitglied (K1,Kollegen).
(2) weisungsberechtigt (K1,K2):–
chef_von (Kollegen,K2),
mitglied (K3,Kollegen),
weisungsberechtigt (K1,K3).

Deklarativ sind obige Regeln wie folgt interpretierbar:
Ein Kollege K2 ist gegenüber einem Kollegen K1 weisungsberechtigit, *wenn*
(1) K2 Chef einer Menge von Kollegen (in Prolog als Liste notiert) ist
und
K1 Mitglied in dieser Liste ist
oder
(2) K2 Chef einer Liste von Kollegen ist
und
ein Kollege K3 Mitglied in dieser Liste ist
und
der Kollege K3 weisungsberechtigt gegenüber dem Kollegen K1 ist.

Einer solchen Interpretation kann eine *prozedurale* Interpretation gegenübergestellt werden:

(1) **(1.1)** Es wird die Abarbeitung der Unterprozedur „chef_von (Kollegen,K2)" versucht. Gelingt dies, so wird die Abarbeitung unter Übernahme der gefundenen Variablenbelegung mit (1.2) fortgesetzt, anderenfalls (d. h. wenn diese ein Backtrack auslöst) mit (2).
(1.2) Es wird die Abarbeitung der Unterprozedur „mitglied (K1,Kollegen)" versucht. Gelingt dies, so ist die Abarbeitung erfolgreich beendet (Antwort „ja" bzw. eine Variablenbelegung). Anderenfalls wird ein alternativer Abarbeitungsweg für (1.1) versucht.
(2) **(2.1)** Es wird die Abarbeitung der Unterprozedur „chef_von (Kollegen,K2)" versucht. Gelingt dies, so wird die Abarbeitung mit (2.2) fortgesetzt. Anderenfalls ist die Abarbeitung erfolglos beendet (Antwort „nein").
(2.2) Es wird die Abarbeitung der Unterprozedur „mitglied (K3,Kollegen)" versucht. Gelingt dies, so wird die Abarbeitung mit (2.3) fortgesetzt. Anderenfalls wird ein alternativer Abarbeitungsweg für (2.1) versucht.
(2.3) Es wird die Abarbeitung der Unterprozedur „weisungsberechtigt (K1,K3)" versucht. Gelingt dies, so ist die Abarbeitung erfolgreich beendet. Anderenfalls wird ein alternativer Abarbeitungsweg für (2.2) versucht. (Wegen der Rekursivität ist die Prozedur Unterprozedur von sich selbst, jedoch mit anderen Aufrufparametern.)

Die Unterprozeduren werden jeweils auf die gleiche Weise abgearbeitet.
Bei der *deklarativen Interpretation* stehen die prädikatenlogischen Aussagen im Vordergrund. Aufgrund der Kommutativität der UND- und ODER-Verknüpfung spielt die Reihenfolge der Klauseln in der Wissensbasis sowie die Reihenfolge der Teilziele in Klauselkörpern hierbei keine Rolle.
Für die prozedurale Interpretation hingegen ist die Reihenfolge, in der die Teilziele durchlaufen und Abarbeitungswege zu deren Erfüllung versucht werden, von Bedeutung. Aus dieser Sicht können die in der Wissensbasis definierten sowie die eingebauten Prädikate auch als Prozeduren und die Teilziele eines Ziels als Prozeduraufrufe interpretiert werden. Insbesondere bei Anwendung von Teilzielen mit „prozeduralen Seiteneffekten" (z. B. solche zur dynamischen Veränderung der Wissensbasis oder zur Ein-/Ausgabe)

muß man sich auch über deren prozedurale Interpretation im klaren sein, um sie anwenden zu können. So ist z. B. das Teilziel „assert(chef_von([krause,albrecht],meyer))", welches das Hinzufügen der als Argument stehenden Klausel zum Programm bewirkt, stets „wahr" und deshalb für die deklarative Interpretation ohne Bedeutung. Ein „prozedurales Mitdenken" ist bei der Verwendung derartiger Prädikate jedoch unumgänglich, da das weitere Programmverhalten durch ihre „Seiteneffekte" entscheidend verändert werden kann.

Steuerung der Abarbeitung

In Prolog ist die Abarbeitung der Fragen fest eingebaut (Tiefensuche mit Backtrack) /1/. Prolog bietet jedoch auch Möglichkeiten, die Abarbeitungsfolge in gewisser Weise zu verändern. Hierzu dienen z. B. die Prozeduren „!" (cut) und „fail".

„!" ist ein stets erfüllbares Teilziel. In Klauselkörpern eingefügt, verhindert es ein Backtrack der hinter „!" stehenden Teilziele zu den vor „!" stehenden Teilzielen sowie zu alternativen Klauseln mit gleichem Kopfprädikat. „!" schneidet die noch verbleibenden alternativen Lösungswege innerhalb der Prozedur, in der es vorkommt, ab. Ein häufig genutzter Nebeneffekt von „!" besteht darin, daß bei dessen Abarbeitung der Speicherplatz für diejenigen Entscheidungspunkte (place marker) und ehemaligen Abarbeitungszustände (Variablenbelegungen) zurückgewonnen wird, zu denen von nun an kein Backtrack mehr möglich ist.

„fail" ist ein nie erfüllbares Teilziel. In Klauselkörpern eingefügt, löst es daher ein Backtrack aus.

Die Anwendung von „!" und „fail" soll an folgendem Beispiel demonstriert werden:

(1) raucher(steffen).
(2) freunde(frank,X):–
 raucher(X),
 !,
 fail.
(3) freunde(frank,X).

Betrachten wir hierzu die Abarbeitung der Ziele

?-freunde(frank,steffen).

(Bild 1) und

?-freunde(frank, rainer).

(Bild 2) am Suchbaum.

Auf das Ziel

?-freunde(frank,steffen).

(Bild 1) sind zunächst zwei Klauseln alternativ anwendbar: (2) und (3). Die Unifikation mit (2) führt zum aktuellen Ziel

?-raucher(steffen),!,fail.

Durch die Anwendung von (1) auf das erste Teilziel kann dieses aus dem aktuellen Ziel entfernt werden, so daß

?-!,fail.

verbleibt. Die Abarbeitung von „!" hat nun zur Folge, daß die alternativ mögliche Anwendung von (3) auf das ursprüngliche Ziel „vergessen" wird. Es verbleibt das immer falsche Teilziel „fail" im aktuellen Ziel, welches zur Beantwortung der Frage mit „nein" führt.

Bei der Abarbeitung des Ziels

?-freunde(frank,rainer).

(Bild 2) wird zunächst der gleiche Weg eingeschlagen. Da jedoch bereits bei der Abarbeitung des Teilziels „raucher(rainer)" ein Backtrack ausgelöst wird, kommt „!" nicht zur Anwendung. Der noch verbleibende alternative Lösungsweg wird eingeschlagen und führt zum Erfolg, d. h. die Frage kann mit „ja" beantwortet werden.

Die sog. „cut-fail-Kombination" verwendet man typischerweise, wenn diejenigen Prämissen gut formuliert werden können, für die ein Prädikat nicht wahr sein soll. Dies ist auch in unserem Beispiel der Fall, da die Wissensbasis keine Fakten vom Typ „nichtraucher(...)" enthält.

Prolog-Fallen
Nicht terminierende Programme

Bereits im Abschnitt 4 /1/ wurde darauf verwiesen, daß das durch Prolog verwendete Tiefensuch-Verfahren mit Backtrack (auch bei im Prinzip vorhandener Lösung) nicht zwangsläufig terminiert. Ein Beispiel hierfür sei folgendes kleine Programm:

(1) vater_von(Vater,Sohn):–
 sohn_von(Sohn,Vater).
(2) sohn_von(Sohn,Vater):–
 vater_von(Vater,Sohn).
 ?-vater_von(ralf,dieter).

Der (entartete) Suchbaum ist im Bild 3 dargestellt. Die Abarbeitung dieses Programmes endet nie. Auch der Speicher läuft nicht über, da keine Entscheidungspunkte und Abarbeitungszustände gekellert werden müssen (es gibt für keines der Prädikate alternativ anwendbare Klauseln, siehe Abschnitt 4 in /1/) und die Anzahl der Teilziele im aktuellen Ziel nicht größer wird.

Außer dem Fall, daß sich ein während der Abarbeitung schon einmal aufgetretenes aktuelles Ziel wiederholt, gibt es noch den Fall, daß das aktuelle Ziel mit jedem Resolutionsschritt immer mehr Teilziele erhält:

(1) teil_von(lampe,elektrik).
(2) teil_von(elektrik,kabel).
(3) teil_von(Teil,Teilesteil):–
 teil_von(Teil,X),
 teil_von(X,Teilesteil).
 ?-teil_von(kabel,draht).

Im Bild 4 ist der Suchbaum hierfür dargestellt. Es wird mit jedem Resolutionsschritt auf das erste Teilziel die Klausel (3) angewandt und dieses durch zwei Teilziele ersetzt. Das aktuelle Ziel wird so mit jedem Resolutionsschritt um ein Teilziel länger, so daß das Programm irgendwann mit einem Speicherüberlauf endet.

Verwendung des logischen „NOT"

In Prolog wird ein 1stelliges Prädikat „not(X)" unterstützt (i. allg. eingebaut, kann aber auch selbst definiert werden mit Hilfe der „cut-fail-Kombination"), dessen Argument die Syntax einer Atomformel hat. Das Teilziel „not(X)" ist erfolgreich, wenn das Teilziel „X" erfolglos ist. Ein einfaches Beispiel demonstriere die Anwendung:

(1) fleissig(dunja).
(2) fleissig(heiko).
(3) faul(X):–
 not(fleissig(X)).

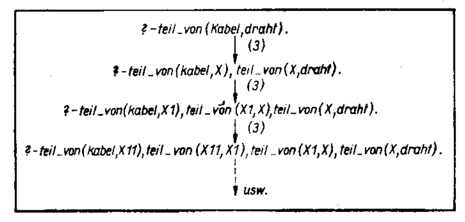


Bild 4
Suchbaum bei immer größer werdender Anzahl der Teilziele im aktuellen Ziel

Stellt man nun beispielsweise die Frage

?-faul(uwe).

, so erhält man die Antwort „ja". Problematisch ist hierbei allerdings die Verwendung von Variablen. Fragt man z. B.

?-faul(Wer).

, so erhält man die Antwort „nein", was offensichtlich im Widerspruch zu obiger Antwort steht. Dieser Effekt rührt daher, daß die Menge der Individuensymbole unendlich mächtig ist und somit Prolog natürlich nicht in der Lage ist, die Komplementärmenge zur Menge {dunja,heiko} zu bilden, um daraus ein Individuensymbol zu entnehmen und es als erste gefundene Lösung zu o. g. Frage anzugeben.

Die im Abschnitt Prolog-Fallen aufgezeigten Effekte sind natürlich nicht immer so offensichtlich wie in den angegebenen kleinen Beispielen und sind deshalb eine häufige Fehlerquelle.

Ein Anwendungsbeispiel

Als geeignetes Einsatzgebiet für die Programmiersprache Prolog erweisen sich wissensverarbeitende Systeme, in denen sich das Fachwissen gut in Form von Hornklauseln notieren läßt. International wird Prolog zunehmend als Implementierungssprache für Expertensysteme eingesetzt. Die Erfahrungen der Autoren am nachfolgend beschriebenen Beispiel belegen ebenfalls die Eignung logischer Programmiersprachen für derartige Anwendungen.

Es wurde ein Programmsystem erstellt, welches die Fehlersuche in technischen Geräten unterstützt. Das zur Diagnose angewandte Wissen ist hierbei explizit in Wissensbasen abgelegt, die

– je nachdem, was für ein Gerät gerade diagnostiziert werden soll, ausgetauscht werden können und
– durch den Nutzer auch ohne große Programmierkenntnisse selbst erstellt und inkre-

Suchbaum bei der Abarbeitung von „?-freunde(frank,rainer)."

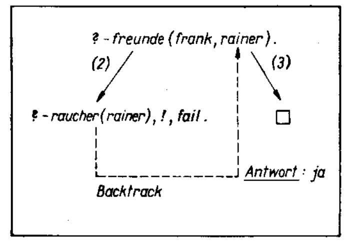


Bild 2

Suchbaum bei der Abarbeitung von „?-freunde(frank,steffen)."

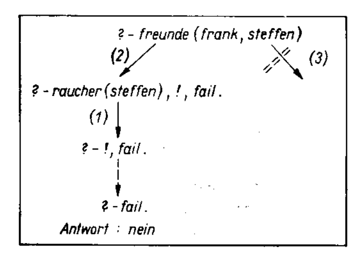


Bild 1

Suchbaum bei wiederholtem Auftreten eines aktuellen Ziels

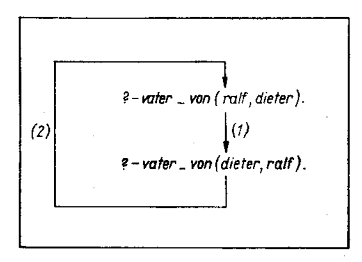


Bild 3

mentell verbessert werden können. Das System ist in der Lage, Wissensinhalte, die nur implizit in der Wissensbasis enthalten sind, nutzbar zu machen und seine Vorgehensweise bei der Fehlersuche zu dokumentieren. In die Wissensbasis wurde zunächst der hierarchische Aufbau des zu diagnostizierenden Gerätes aufgenommen. Als Darstellungsmittel auf Nutzerebene schlagen die Autoren ein semantisches Netz vor, dessen Knoten die Funktionsgruppen eines Gerätes symbolisieren und dessen gerichtete Kanten mit der Relation „teil_von" markiert sind. Dieses Netz weist eine Baumstruktur auf. Eine solche Darstellung läßt sich sehr einfach in eine Prolog-Darstellung überführen. Man könnte beispielsweise jeden (nicht terminalen) Knoten des Baumes zusammen mit der Liste seiner Nachfolger in je einem Fakt der Form

teile_von(<Teil,>,[<Teil>],...,<Teil>]).

notieren.

Als nächstes müssen jeder Funktionsgruppe logisch verknüpfte (UND, ODER, NICHT) Symptome zugeordnet werden, die hinreichend dafür sind, daß ein Fehler in dieser Funktionsgruppe liegt. Eine jede solche logische Verknüpfung kann ohne Beschränkung der Allgemeinheit als disjunktive Normalform

$$\bigvee_{i \le n} \bigwedge_{j \le m} \langle Symptom \rangle_{ij}$$

notiert werden. Die Darstellung der Symptome, der logischen Verknüpfung und der Zuordnung zu den Funktionsgruppen konnte in Prolog wie folgt gelöst werden (vereinfacht):

Jedem Symptom (welches durch einen auszugebenden Text charakterisiert ist) wurde ein Kodename zugeordnet und diese Zuordnung als Fakt der Form

symptom(<Kode>,<Text>).

notiert. Die logische Verknüpfung der Symptome in disjunktiver Normalform und deren Zuordnung zu den Funktionsgruppen konnte dargestellt werden als

symptome_von(<Fkt_gr>,[<Fk>,...,
<Fk>]).

, wobei <Fk> Fundamentalkonjunktionen (UND-verknüpfte Symptome) der Form

[<Symptom>,...,<Symptom>]

sind , die ihrerseits ODER-verknüpft sind. Das Ziel der Fehlersuche besteht nun darin, in dem den Aufbau des Gerätes repräsentierenden semantischen Netz („teil_von – Baum") einen Pfad von der Wurzel (Knoten ohne Vorgänger) bis zu einem terminalen Knoten (Knoten ohne Nachfolger) zu suchen. Der Pfad hat die Eigenschaft, daß für jeden auf diesem Pfad liegenden Knoten die Symptome bestätigt sind. Als Suchverfahren wurde eine Tiefensuche mit Backtrack im „teil_von – Baum" gewählt, die sich in Prolog recht einfach programmieren läßt:

```
diagnostiziere(Teil):–
    teile_von(Teil,Teileliste),
    mitglied(Teilesteil,Teileliste),
    symptome_bestaetigt(Teilesteil),
    diagnostiziere(Teilesteil).
```

Gelingt es nicht, die Symptome für ein „Teilesteil" (indirektes Teil eines Teiles) zu bestätigen, so sorgt der Backtrack-Mechanismus von Prolog dafür, daß eine solche Bestätigung mit dem nächsten Mitglied der Teileliste versucht wird. Gelingt dieses für ein „Teilesteil", so wird durch den rekursiven Selbstaufruf der Prozedur die Suche ab dort nach dem gleichen Verfahren fortgesetzt.

Um die Symptome für ein Teil zu bestätigen, genügt es, eine der Fundamentalkonjunktionen zu bestätigen:

```
symptome_bestaetigt(Teil):–
    symptome_von(Teil,Symptome),
    mitglied(Fund_konj,Symptome),
    bestaetigt(Fund_konj).
```

Um eine Fundamentalkonjunktion zu bestätigen, müssen alle in ihr enthaltene Symptome bestätigt werden:

```
bestaetigt([]).
bestaetigt([Symptom|Restliste]):–
    bestaetige(Symptom),
    bestaetigt(Restliste).
```

Die Realisierung von „bestaetige(Symptom)" ist Bestandteil einer speziellen Dialogkomponente, die hier nicht weiter betrachtet werden soll.

Obwohl die Realisierung des Systems hier nur angedeutet werden konnte, sieht man doch deutlich, daß Prolog für dessen Erstellung ein geeignetes Werkzeug ist.

Literatur

/1/ Knauf,R.; Killenberg, H.: Programmieren in PROLOG. Mikroprozessortechnik, Berlin 1 (1987) 11, S. 339

/2/ Knauf, R.: Untersuchungen zu Aufbau und Implementierung von Expertensystemen und Realisierung eines Diagnosesystems in der Programmiersprache PROLOG. Forschungsbericht, TH Ilmenau, Sektion TBK

/3/ Killenberg, H.; Knauf, R.; Schumann,, H.-J.: Künstliche Intelligenz. Projektionsfolienreihe, Institut für Film, Bild und Ton, Reg.-Nr. HFR 742, Berlin 1988

✉ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Technische Informatik,
PSF 327, Ilmenau, 6300; Tel. 74 561

Experimentalsystem für Mini-MAP

Maike Prager, Hellfried Schumacher
Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik

MAP-Projekt
Internationaler Stand

Der internationale Trend zur schrittweisen Automatisierung in der Produktion führte in den letzten Jahren zur Herausbildung von konzentrierten Automatisierungsinseln. Erst relativ spät wurde erkannt, daß für eine vollständige, computergesteuerte Fertigung eine Kommunikation zwischen diesen Inseln erforderlich ist. Bedingt durch die Inkompatibilität der Rechner und Geräte vieler Hersteller erweist sich deren nachträgliche Vernetzung als zu aufwendig und teuer. Deshalb sah man sich gezwungen, ein gemeinsames Projekt zur Vereinheitlichung der Datenkommunikation in der automatisierten Fertigung durchzuführen, um die Marktanteile für eigene Geräte weiterhin zu sichern.

Das Ergebnis dieser Bemühungen liegt seit wenigen Jahren in Form des Manufacturing Automation Protocol (MAP) vor und beinhaltet eine geeignete Auswahl bereits vorhandener Standardisierungsprotokolle aus dem Bereich der Lokalen Rechnernetzwerke (LAN) sowie die Definition noch fehlender Vereinbarungen.

Gleichzeitig mit dem sogenannten MAP-Backbone, das für die Vernetzung aller automatisierungstechnischen Anteile einer Produktion verantwortlich ist, wurde für den bürotechnischen Bereich (Verwaltung, Auftragsentgegennahme, Absatz) das TOP-Projekt entwickelt. Dieses unterscheidet sich natürlich Echtzeitverhalten (Zugriff) bis zur Anwendungsschicht vom MAP.

Der Umfang und die Tragweite des MAP-Projekts spiegelt sich auch im Entwicklungszeitraum wider, der 1981 mit einem Konzept, von General Motors initiiert, begann und dessen stufenweise Verwirklichung sich heute in der aktuellen Version MAP 3.0 niederschlägt. Die trotz enormer Mittel lange Bearbeitungsdauer, die dieses Protokoll vom Konzept bis zur Verwirklichung benötigte, sowie die weltweite Resonanz rechtfertigen wohl eindeutig, daß ein anderer, eigener Entwurf eines solchen Vorhabens die vorhandenen zeitlichen sowie personellen Mittel bei weitem übersteigen würde. Zusätzlich muß man damit rechnen, daß zukünftig die Interfaces der auf dem internationalen Markt erhältlichen Geräte diesem Standard gerecht werden.

MAP ist, ebenso wie andere Standards für Rechnernetze, nach dem OSI-Modell strukturiert. Dieses beinhaltet sieben sogenannte Schichten mit spezifischen Funktionen, z. B. Kodierung, Datenauskopplung, Mediumzugriff, Datensicherung usw. bis zu anwenderspezifischen Aufgaben. Jede Schicht besitzt ein eigenständiges Protokoll und definierte Schnittstellen zum Informationsaustausch mit den benachbarten Schichten. Auf eine genaue Beschreibung dieses Modells soll an dieser Stelle mit Hinweis auf /1/ und /2/ verzichtet werden. Tafel 1 gibt an, welche Standards im Rahmen von MAP für die einzelnen OSI-Schichten vorgeschrieben werden.

Die Implementierung aller sieben Schichten des OSI-Modells im MAP-Backbone hat, bedingt durch Overheads, die durch schichtenspezifische Protokollinformationen entstehen, eine Verringerung der Nettodatenrate sowie einen geringeren Datendurchsatz durch die Schichten zur Folge. Das heißt, trotz echtzeitfähigen, deterministischen Mediumzugriffs und hoher Datenübertragungsrate ist das Backbone für Automatisierungslösungen mit harten Echtzeitanforderungen bzw. für Insellösungen, also Produktionszellen, nicht anwendbar.

Diese Forderungen haben schon sehr kurz nach dem Entwurf des MAP-Konzepts ihren Niederschlag in der sogenannten Mini-MAP-Architektur bzw. in der MAP-EPA (Enhanced Performance Architecture) gefunden.

Tafel 1 Internationale Standards für die einzelnen OSI-Schichten

OSI-Schicht	MAP-Backbone	Mini-MAP
7 (Anwendung)	MMS (RS-511) ISO FTAM	MMS (RS-511) ISO FTAM
6 (Darstellung)	ISO PRESENTATION	NULL
5 (Sitzung)	ISO SESSION Kernel	NULL
4 (Transport)	ISO Transport Class 4	NULL
3 (Vermittlung)	ISO INTERNET Protocol	NULL
2 (Verbindung) LLC MAC	IEEE 802.2 Type 1 IEEE 802.4 Token Bus	IEEE 802.2 Type 3 IEEE 802.4 Token Bus
1 (Übertragung)	IEEE 802.4 Broadband (10 MBit/s)	IEEE 802.4 Carrierband (5 MBit/s)

Bei diesen Architekturen wird durch Auslassen der OSI-Schichten 3 bis 6 (vergleiche Tafel 1) der Overdead reduziert, was zu einem Nettodatendurchsatz von etwa 250 KBit/s im Gegensatz zum Backbone (120...150 KBit/s) führt (Angaben laut /3/, /4/). Der Netzwerktyp des Mini-Map entspricht dem des Backbone, das heißt, das Token-Passing-Verfahren und die Busstruktur bleiben erhalten (siehe Abschnitt *Übertragungsschicht)*.
Dieses Netzwerk ist als weitaus kostengünstiger und schneller realisierbar anzusehen.
Bei der Konzeption der Arbeit an der Sektion Technische Elektronik, die vor etwa 2 Jahren begann, ließen wir uns von diesem Mini-MAP leiten und versuchten, dabei eigene Formen der Realisierung zu finden. Da die EPA-Form aber kompatibel zum Backbone ist, läßt sich darauf aufbauen.

Übertragungsschicht

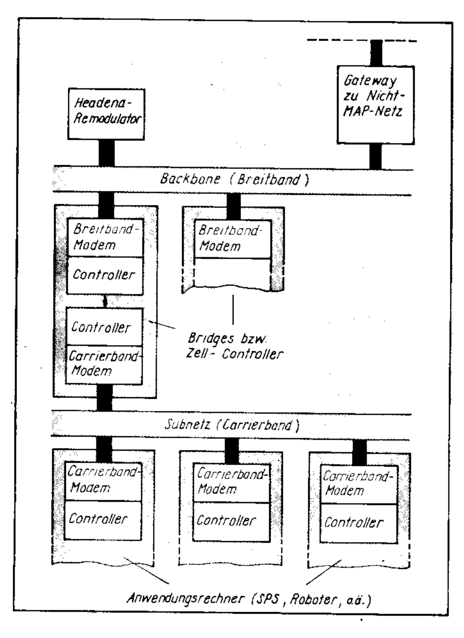
Es gibt zwei unterschiedliche Techniken zum Aufbau eines MAP-gerechten Token-Bus-LAN: Breitband und Carrierband (Tafel 2). Diese sind im IEEE-Standard 802.4 als *Broadband Bus Physical Layer* bzw. *Single-Channel Phase-Coherent FSK Bus Physical Layer* definiert /5/. Beide verwenden 75-Ohm-Koaxialkabel als Übertragungsmedium. Es gibt außerdem Bestrebungen, eine dritte Variante, die auf der Verwendung von Lichtwellenleitern basiert, zu standardisieren /6/.

Tafel 2 Carrierband und Breitband

	Carrierband	Breitband
Datenrate	*5 MBit/s*	*(5 MBit/s), 10 MBit/s*
Anzahl der Kanäle	*1*	*je nach verfügbarer Kabelbandbreite*
Modulationsverfahren	*phasenkohärente Frequenzumtastung*	*zweifach binäre Amplitudenmodulation/Phasenumtastung*
Übertragungsmedium	*75-Ohm-Koax-Kabel*	*75-Ohm-Koax-Kabel*
Verkabelung	*ähnlich CATV (passive Komponenten)*	*ähnlich CATV (mit Frequenzumsetzer)*
Signalübertragung	*bidirektional*	*unidirektional*

Das Breitband-Verfahren beruht auf der Unterteilung der verfügbaren Kabelbandbreite in mehrere Übertragungskanäle, wobei Sende- und Empfangskanäle unterschieden werden, da die Übertragung innerhalb eines Kanals nur in einer Richtung erfolgt. Zur Signalumsetzung zwischen Sende- und Empfangskanälen dienen Frequenzumsetzer (Head-End-Remodulator). Dieses Verfahren ermöglicht auch die Einrichtung mehrerer Netze auf einem Kabel. Da die Festlegung des Standards entsprechend den Normen des Kabelfernsehens (Community-Antenna Television, CATV) erfolgte, sind die hierfür entwickelten Übertragungssysteme verwendbar. Gegebenenfalls können freie Kanäle eines CATV-Systems genutzt werden. Wegen der erforderlichen aktiven Übertragungssysteme (Frequenzumsetzer) ist die Installation eines Breitband-LAN sehr material- und kostenaufwendig. Im Rahmen von MAP sind Breitbandnetze als fabrikübergreifende Backbone- (Rückgrat-) Netze vorgesehen.
Für kostengünstigere Realisierungen sieht der MAP-Standard das Carrierbandverfahren vor. Ein Carrierbandsystem hat nur einen Kanal. Alle Teilnehmer senden und empfangen auf einer Frequenz. Im Unterschied zur verbreiteten Basisbandübertragung wird das Signal jedoch moduliert. Die Übertragung auf dem Medium erfolgt bidirektional, es sind also keine Remodulatoren oder ähnliches erforderlich. Carrierbandnetze werden im Rahmen von MAP als Subnetze zur Verbindung von programmierbaren Steuerungen, Robotern, CNC-Maschinen und ähnlichem innerhalb einer Produktionszelle Verwendung finden. Die Verbindung dieser Netze mit dem übergeordneten Backbone erfolgt durch sogenannte Bridges (vergleiche Bild 1). Dabei ist auch eine Kombination von Bridges und Funktionen zur Steuerung und Koordinierung der Fertigungszelle in einem sogenannten Zell-Controller möglich /3/.

Bild 1 MAP-Backbone und Subnetz



Mediumzugriff

Der Mediumzugriff wird durch die Medium-Access-Control-Subschicht (MAC) realisiert. Diese bildet den unteren Teil der Verbindungsschicht (Schicht 2 des OSI-Modells).
Im MAP-Standard wird der Token-Bus-Algorithmus nach IEEE 802.4 /5/ vorgeschrieben. Dieser setzt als Netzwerktopologie (Struktur) den Bus voraus. Auf diesem Bus wird zwischen den Stationen ein logischer Ring aufgebaut, auf dem das Token und mit ihm das Zugriffs- und Senderecht ständig kreist. Dieser Ring ist dynamisch, das heißt, nicht alle Stationen müssen ständig im Ring sein, können aber in periodischen Abständen in den Ring aufgenommen werden bzw. ihn zu jeder Zeit (bis auf den aktuellen Token-Besitzer) verlassen. Dabei ist nicht zwingend, unbedingt Sendedaten zu haben, um Ringmitglied zu sein. Das ist besonders für Stationen mit seltenen aber dringenden Sendungen vorteilhaft und wird mit Hilfe von Managerparametern des Netzwerkmanagers verwirklicht. Im statischen, also unveränderten, Zustand ist dieser logische Ring mit dem Token-Ring vergleichbar, wenn man von der Quell- und Zieladressierung absieht.
Die Stationseinfügung wird vom aktuellen Token-Besitzer zu festlegbaren Zeitpunkten mit Hilfe unterschiedlichster Steuerframes realisiert und gestattet Stationen, deren Adressen sich zwischen Ziel- und Quelladresse des Tokens befinden, in den Ring zu gelangen.
Bewerben sich mehrere Stationen gleichzeitig um eine Ringaufnahme, wird dieser Mechanismus allerdings relativ zeitaufwendig und kompliziert, da der Token-Besitzer eine Station auswählen muß. Dafür wurde der sogenannte „Response-Window-Process" geschaffen, ein Prozeß, bei dem nach entsprechender Reaktion des Token-Besitzers die anfordernden Stationen entsprechend ihren Adreßbits unterschiedlich reagieren und somit nach Prozeßende nur eine Station übrigbleibt.
Der Token-Bus-Algorithmus garantiert die Behebung solcher Fehler wie Verlust oder Mehrfachauftreten eines Tokens. Die Station, die das Token besitzt, ist in der Lage, Datensendungen zu übertragen. Die Anzahl der übertragenen Datenframes, die maximal 8 KByte lang sind, richtet sich nach der sogenannten Tokenhaltezeit, das heißt die Zeit, die einer Station zur Sendung zur Verfügung steht.
Um das Token sicher weiterzureichen, wird der Bus nach Weitergabe für eine bestimmte Zeitdauer abgehört und bei Bedarf ein neuer Nachfolger bestimmt.
Bei Implementierung des geforderten Prioritätsalgorithmus wird mit Unterstützung von prioritätsspezifischen Timern eine entsprechende Priorisierung der Sendungen erreicht.
Dieser Algorithmus ermöglicht auch, daß der Tokenumlauf unterbrochen wird und das Token zu Stationen mit höherpriorisierten Sendungen umgeleitet wird.
Nach Abarbeitung aller zeitkritischen Sendungen wird das Token wieder zur unterbrochenen Station zurückgereicht.
Als letztes soll der Hinweis genügen, daß sowohl Broadcastsendungen (an alle Stationen) als auch Multicastsendungen (an bestimmte Zielgruppen) in der Adressierung vorgesehen sind. Die Länge der Adresse wurde vom MAP-Standard auf 6 Byte festge-

legt. Die Behandlung der eigentlichen Nutzerdaten erfolgt durch die MAC-Zwischenschicht nicht. Diese Nutzerdaten werden sofort zur Datensicherungsschicht (LLC) weitergereicht, sofern sie fehlerlos, das heißt mit gültigem FCS empfangen worden sind.

Datensicherungsschicht

Der MAP-Standard schreibt, wie schon erwähnt, als Datensicherungsschicht (Logical-Link-Control, LLC) den verbindungslosen Dienst des Standards IEEE 802.2 /7/ vor. Aufgabe dieser Schicht ist es, die empfangenen Nutzdaten zu behandeln und die zu sendenden Daten für die Sendung vorzubereiten, indem unter anderem die MAC über einen notwendigen Zugriff informiert wird. Der verbindungslose Dienst ist dabei im Gegensatz zum verbindungsorientierten Dienst, der ebenfalls Bestandteil des IEEE 802.2 ist, die einfachste Realisierungsvariante und demzufolge auch die mit dem schnellsten Datendurchsatz durch die zweite Schicht.

Es handelt sich dabei um ein unquittiertes Protokoll, das heißt, nach Erlangen des Netzwerkzugriffs wird zunächst durch entsprechende Steuerframes mit der Zielstation eine Verbindung aufgebaut und von dieser auch bestätigt. Nach Aufbau dieser Verbindung werden die eigentlichen Datenframes ohne Bestätigung und unnumeriert hintereinander übertragen, solange es die Tokenhaltezeit erlaubt.

Es ist offensichtlich, daß diese Art der Datenübertragung mit einem verhältnismäßig geringem Overhead durch wenig LLC-Steuerframes belastet wird. Es muß aber bemerkt werden, daß bei mehreren zusammengehörenden Datenframes der Verlust eines Frames zumindest durch die LLC-Schicht nicht bemerkt wird, so daß die nächst höhere Schicht für eine entsprechende Datensicherung Sorge tragen muß. Die Funktionen der Fehlererkennung und Rückgewinnung müssen also durch das Protokoll der Schicht 4 (Transportschicht) übernommen werden. Da das Mini-MAP-Protokoll aber auf die Schichten 3 bis 6 verzichtet, mußten hier andere Formen der Datensicherung gefunden werden.

Der verwendete LLC-Dienst Typ 3 des Standards IEEE 802.2 wird diesen Forderungen gerecht, indem er Funktionen des ANSI-Proway-Standards beinhaltet, die wahlweise eine Datensendung mit Quittierung (SDA, send data with acknowledge service) bzw. eine Datenanforderung mit Wiederholungen (RDR, request data with reply service) gestatten /8/, /9/. Diese Funktionen sind als Option anzusehen, da in den meisten Fällen der Automatisierung unter Echtzeitbedingungen sehr häufig und hierbei relativ kurze Parameter oder Resultate übertragen werden, die in nur einem Datenframe (maximal 8 KByte) untergebracht werden können.

Der Verlust nur eines Datenframes wird aber bemerkt, da nach Verbindungsaufnahme innerhalb einer festgelegten Zeit mindestens ein Datenframe erwartet wird und deshalb reagiert werden kann.

Anwenderschicht

Bei der Implementierung der Applikations- oder auch Anwenderschicht ließen frühere MAP-Versionen die meisten Freiheiten zu, was einen für den Anwender größeren Implementierungsaufwand zur Folge hatte. Die letzte Version MAP 3.0. bietet aber eine anwenderfreundlichere Schnittstelle, sowohl für den Filetransfer als auch für Automatisierungsprotokolle, die unter anderem nach den Standards FTAM und RS-511 realisiert werden /10/.

Für das Mini-MAP entstanden verschiedene Vorstellungen für diese Schicht. Im EPA wird eine vollständige Kompatibilität zum MAP-Backbone mit der Wahl des *Manufacturing Message Service* (RS-511) angestrebt.

Um besonders zeitkritischen Anwendungen Rechnung zu tragen, hat sich aber seit den MAP-Anfängen auch ein Konzept gehalten, das in der aufgesetzten Schicht lediglich ein *Objekt Dictionary System* beinhaltet. Hier würden dann die aktuellen Variablen bzw. Prozeßparameter von CNC-Maschinen, programmierbaren Steuerungen und ähnlichen abgespeichert und somit ständig abrufbar sein. Außerdem muß natürlich eine entsprechende Schnittstelle zur LLC-Schicht realisiert werden. Solch ein Konzept mit minimalem Overhead und größtem Datendurchsatz wäre für viele Automatisierungslösungen sicher akzeptabel.

Es wäre allerdings auch denkbar, einen Filetransfer zu implementieren, um z. B. zu Beginn einer Automatisierungsaufgabe die einzelnen Stationen des Systems zu initialisieren bzw. neu zu programmieren.

Produkte

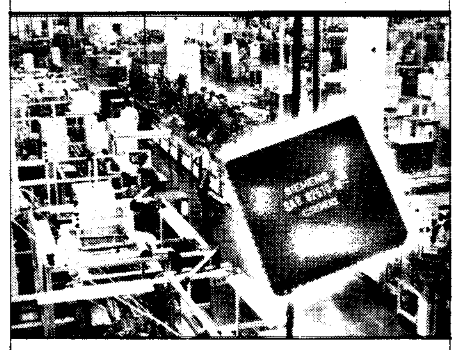
Entsprechend einem Stufenprogramm wurden erste praktische Realisierungen von MAP-Netzwerken ab etwa 1984 vorgestellt. Diese noch sehr kostenaufwendigen Lösungen dienten nicht nur zu Experimentalzwekken, sondern auch zur Demonstration der MAP-Philosophie, das heißt dem Nachweis, daß inkompatible Systeme verschiedener Hersteller über ein gemeinsames Netz kommunizieren können. Als Beispiel sei die sehr werbewirksame Vorstellung eines Modellnetzes auf der Ausstellung *Autofact '85* in Detroit genannt /11/.

Die zu einer kostengünstigen MAP-Implementierung erforderlichen Baugruppen und Systeme werden seit etwa 1986 von einigen, meist US-amerikanischen Herstellern angeboten. Diese Erzeugnisse umfassen spezielle Schaltkreise, Interface-Baugruppen und Software.

Zu den Schaltkreisen gehören Token-Bus-Controller (TBC) für verschiedene Mikroprozessorfamilien, z. B. MC 68824 von Motorola /12/ und SAB 82510 von Siemens (siehe Kasten). Das sind VLSI-Schaltkreise, die sämtliche Funktionen der MAC-Subschicht entsprechend IEEE 802.4 auf einem Chip realisieren (Bild 2). Außerdem wurden verschiedene Interface-Schaltkreise für die digitalen Funktionen der physikalischen Schicht entwickelt, z. B. von Motorola der Broadband-Interface-Controller (BIC) MC 68184 und das Carrierband-Modem (CBM) MC 68194 sowie von Siemens das Carrierband-Modem SAB 82511.

Solche Schaltkreise bilden die Grundlage für verschiedene Interface-Leiterkarten, die als OEM-Baugruppen für die international verbreiteten Bussysteme angeboten werden, z. B. MVME372 von Motorola /12/, iSXM554 von Intel /13/, Serie 1200 von Concord Communications /14/ und MM-400, MQ-400, MV-400, MP-500 von Industrial Networking Incorporated. Zusammen mit entsprechenden Modem-Baugruppen gestatten diese Controller-Leiterkarten eine relativ einfache und kostengünstige vollständige MAP-Implementierung. Die Protokolle der höheren OSI-

MAP-Controller auf einem Chip



Für Manufacturing Automation Protocol (MAP) bietet Siemens jetzt auch den Token-Bus-Controller SAB 82510 an (siehe Bild). Zusammen mit einem vor Jahresfrist vorgestellten MAP-Modem (SAB 82511) bildet der CMOS-Controller in der automatisierten Fabrik der Zukunft die Schnittstelle zwischen dem Systembus in den Geräten bzw. Anlagen und dem Kabel des Token-Bus, der sämtliche Stationen eines derartigen Netzwerkes verbindet. Der neue Token-Bus-Controller integriert die komplette Hardware von Schicht 2a des OSI-Modells, für die bisher eine aufwendige Baugruppe im Format von etwa 200 × 160 mm benötigt wurde. Das MAP-Modem integriert OSI-Schicht 1.

Der neue Token-Bus-Controller von Siemens entspricht IEEE 802.4 als der international verbindlichen Richtlinie für die Vernetzung der Fabrik der Zukunft. Die aufwendige Logik des Bausteins wird mit rund 100 000 Transistorfunktionen bewältigt. Für den Datentransfer über Koaxialkabel von wahlweise 1 oder 5 MBit/s sowie 10 MBit/s verfügt der Chip des MAP-Controllers SAB 82510 über einen eigenen DMA-Controller. Eine Transferrate von 20 MBit/s für Glasfaserbetrieb wird vorbereitet.

Auf dem Wege zur effizienten Fabrik der Zukunft werden die beiden MAP-Bausteine als Meilensteine angesehen. Über den Vorteil der Integration bisher individuell aufgebauter Baugruppen hinaus bieten die Einchiplösungen für die OSI-Schichten 1 und 2 eine Voraussetzung, um Geräte bzw. Stationen unterschiedlicher Hersteller komplikationsfrei vernetzen zu können. Die Spezifikationen von IEEE 802.4 sichern einheitliche Interfaces für unproblematischen Datenübergang. Der Wildwuchs bisheriger Individuallösungen in diesem Bereich macht die Vernetzung unnötig aufwendig und kompliziert. Für eine herkömmliche MAP-Controller-Baugruppe muß im Vergleich zu einer Einchiplösung etwa der zehnfache Betrag aufgewendet werden. *MP*

TERMINE

9. Zuverlässigkeitstagung „Qualität und Zuverlässigkeit in automatischen Produktionsprozessen"

WER? Fachverband Elektrotechnik der KDT
WANN? 11./12. Mai 1989
WO? Leipzig
WAS?
- theoretische Grundlagen der Zuverlässigkeitstechnik
- Ausfallmechanismen und Ausfallanalyse
- Fertigungsprozeß und Zuverlässigkeit
- Zuverlässigkeit unter Umgebungsbeanspruchungen
- Qualität der Software

WIE? Teilnahmemeldungen bitte an Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Straße 115–117, Berlin, 1086

Hoppe

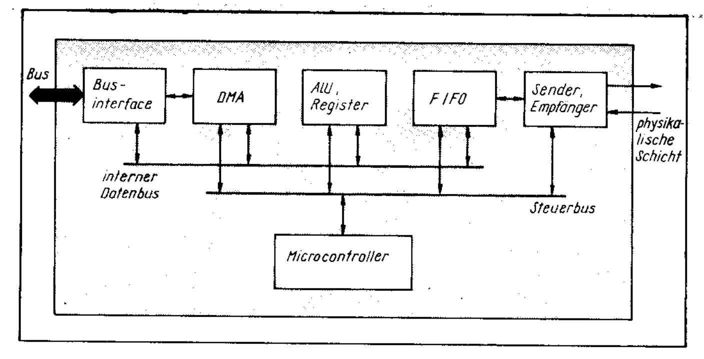


Bild 2 Struktur eines Token-Bus-Controllers (TBC)

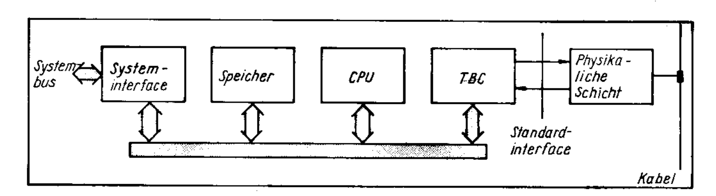


Bild 4 Blockschaltbild der Controllerbaugruppe

◀ Bild 3 MAP-Interface mit TBC

Schichten, also LLC-Subschicht bis Schicht 7, werden softwaremäßig realisiert, was auf der Controller-Leiterkarte sowohl eine leistungsfähige 16- bzw. 32-Bit-CPU als auch eine entsprechende Speicherkapazität erfordert. Bild 3 zeigt ein allgemeines Blockschaltbild solcher Controller. Die Kommunikationssoftware umfaßt entsprechende Routinen für jede OSI-Schicht, die unter einem Echtzeitkern laufen. Soweit angegeben (/12/, /15/, /16/), wurden die Programme in C geschrieben.

Mini-MAP-Experimentalsystem
Hardware

Um eigene praktische Untersuchungen an einem Token-Bus-LAN zu ermöglichen, wurde eine entsprechende Controller-Baugruppe entwickelt. Ausgehend vom derzeit verfügbaren Bauelementesortiment wurde eine Variante gewählt, welche einige Funktionen der physikalischen Schicht und der MAC-Subschicht hardwaremäßig realisiert und die softwaremäßige Implementierung der restlichen MAC-Funktionen (Zugriffsverfahren) ermöglicht.
Kernstück der Baugruppe (vergleiche Bild 4) ist ein Single-Board-Computer (SBC) mit dem Mikroprozessor 8086, eine Weiterentwicklung des in /17/ vorgestellten SBC 8086. Dieser wird ergänzt durch
- ein schnelles serielles Interface mit dem Serial Communication Controller (SCC) U 82530 /18/, der eine Datenübertragung entsprechend SDLC mit maximal 1 MBit/s gestattet
- ein einfaches Modem einschließlich Leitungstreiber und -empfänger
- einen I/O-Koprozessor 8089 zur schnellen Datenübertragung zwischen SCC und Speicher (DMA)
- zusätzliche Timer zur Realisierung der Zeitverläufe innerhalb der MAC
- Dual-Port-RAM zur Datenübergabe an einen übergeordneten Rechner (weiterer SBC), welcher die Funktionen der höheren OSI-Schichten realisiert.

Auf der Grundlage dieses Controllers läßt sich ein Token-Bus-LAN aufbauen, das allerdings nicht vollständig dem Standard IEEE 802.4 entspricht. Abweichungen, z. B. hinsichtlich der elektrischen Parameter, der Übertragungsgeschwindigkeit, Symbolkodierung und Frame-Format, müssen in Kauf genommen werden. Es werden aber keine Einschränkungen bezüglich des Token-Bus-Verfahrens notwendig, das heißt, das LAN wird in Bus-Topologie aufgebaut und der Mediumzugriff entsprechend IEEE 802.4 realisiert.
Mit dem beschriebenen Aufbau sind Untersuchungen zur Leistungsfähigkeit des Token-Bus-Algorithmus möglich. Außerdem bildet er eine Grundlage für die Realisierung der höheren Schichten. Es ist vorgesehen, die Software für die höheren OSI-Schichten auf einem weiteren SBC unterzubringen. Für diese Aufteilung auf zwei eigenständige Rechner sprechen folgende Überlegungen: Erstens wird durch die gleichzeitige Bearbeitung von MAC und der übrigen Schichten die Leistungsfähigkeit des Systems und damit der Datendurchsatz erhöht. Zweitens ist der MAC-Algorithmus relativ rechenaufwendig und stellt sehr hohe Echtzeitanforderungen. Trotz der vergleichsweise niedrigen Datenrate von 1 MBit/s erfordert beispielsweise der *Response-Window-Process* Reaktionszeiten von nur wenigen Mikrosekunden. Diese könnten z. B. bei einer Multitask-Abarbeitung nur schwer garantiert werden. Drittens ermöglicht die Modularität dieses Konzepts gegebenenfalls den Austausch des Controller-SBC durch einen integrierten Token-Bus-Controller.

Software

Das Gesamtkonzept ist modular gestaltet. Jede Schicht setzt sich dabei aus einer Gruppe von Routinen zusammen und wurde mit der Programmiersprache C erstellt.
Kennzeichnend für das OSI-Modell ist die Informationsübermittlung zwischen den Schichten, die durch sogenannte Dienstprimitiven erfolgt. Diese enthalten sowohl Steuerinformationen als auch gegebenenfalls Zeiger auf die Daten. Das Umschalten zur Abarbeitung einer bestimmten Schicht im Anforderungsfall wird zur Zeit mit Hilfe eines Kernprogramms im Polling betrieben, das heißt, je nach Bedarf erhält die entsprechende Schicht CPU-Zeit. Die Dienstprimitiven werden dabei durch Speicheroperationen übergeben.
Um eine möglichst problemlose und quasiparallele Abarbeitung mehrerer Sende- oder Empfangsanforderungen zu garantieren, wird die Einbindung der LLC-Schicht, Applikationsschicht und des Netzwerkmanagers in das Echtzeit-/Multitasksystem BOS 1810 erfolgen. Jede Schicht wird dabei als selbständige Task ausgeführt. Die Kommunikation unter den Schichten erfolgt mit Hilfe von Objekt-Queues (Objekt-Warteschlange), wobei für jede Schicht eine gesonderte Warteschlange vorgesehen ist.
Die Primitivenübergabe zwischen der LLC und der MAC-Subschicht, die aus genannten Gründen nicht unter BOS 1810 läuft, wird über einen Dualport-RAM erfolgen.
Bei der Realisierung des Mediumzugriffs (Token-Bus-Algorithmus) sowie der Datensicherung wurde ausschließlich auf die vorgeschriebenen Standards IEEE 802.4 und IEEE 802.2 sowie auf die genannten MAP-spezifischen Anforderungen Bezug genommen.
Zum Token-Bus-Algorithmus bleibt anzumerken, daß die ständigen dynamischen Veränderungen in der Reihenfolge des Tokenumlaufs natürlich sehr zeitintensiv sind. Deshalb ist auch je nach Netzwerklast, Häufigkeit der Sendeanforderungen, durchschnittlicher Länge sowie Prioritäten der Datensendungen eine sorgfältige Wahl aller dafür verantwortlichen Managerparameter und Zeitkonstanten erforderlich. Es wird aber garantiert, daß durch entsprechende Kombination der Parameter priorisierte Sendungen innerhalb eines Zeitlimits übertragen werden.
Aussagen über die Abhängigkeit der Parameter von den erforderlichen Echtzeitanforderungen sowie über erreichbare minimale Übertragungszeiten werden ein Hauptbestandteil unserer weiteren Untersuchungen werden. Die Implementierung der Datensicherungsschicht (LLC) erfolgte zunächst ohne die im Abschnitt *Datensicherungsschicht* genannte Option einer Datensendung mit Quittierung bzw. einer Datenanforderung mit Wiederholungen, die für das EPA-Mini-MAP bindend ist. In einer späteren Implementierungsphase soll sie berücksichtigt werden, um Aussagen über Vergleiche beider Versionen treffen zu können.
Unsere Lösung sieht vor, für die aufgesetzte Applikationsschicht neben einem im Abschnitt *Anwenderschicht* beschriebenen Objekt-Dictionary-System auch einen Filetransfer zu realisieren.
Da die Schnittstelle zur LLC-Schicht gleich ist, wäre in späteren Zeiträumen auch ein kompletter Austausch durch die Applikationsschicht des EPA-Modells grundsätzlich

möglich. Zusätzlich zu diesen Schichten wird ein Netzwerkmanager realisiert, der es dem Nutzer gestattet, sowohl echtzeitspezifische Parameter für die MAC-Schicht zu verändern als auch statistische Informationen z. B. über den aktuellen Abarbeitungszustand der einzelnen Sendungen innerhalb des Netzwerkes oder über Fehlerquoten zu erhalten.

Literatur

/1/ Heymer, V.: Lokale Rechnernetze mit OSI-Architektur. Mikroprozessortechnik 1 (1987) 3, S. 74
/2/ Loeffler, H.: Lokale Netze. Berlin: Akademie-Verlag 1987
/3/ Segl. E.: Zellenstruktur am MAP-Backbone. Elektronik (1986) 10, S. 128
/4/ MiniMAP-Standard zum Anfassen. industrie-elektrik + elektronik (1987) 8, S. 6
/5/ ISO/Draft International Standard 8802/4, 1984
/6/ Sablehaus, M.: Fiber optics and MAP. I&CS (1986) 11, S. 67
/7/ IOS/Draft International Standard 8802/2, 1984
/8/ Crowder, R. S.: Putting MAP/TOP products into perspective. I&CS (1986) 11, S. 49
/9/ Hollingum, J.: The MAP Report, Manufacturing Automation Protocol. Berlin (W), Heidelberg, New York, Paris, Tokyo: Springer Verlag 1987
/10/ Simon, Th.: Neue MAP-Schnittstellen für die Anwendungsschicht. Hard and Soft (1987) 11, S. 42
/11/ Dieterle, G.: Der Weg zu MAP und TOP. Automatisierungstechnische Praxis (1986) 4, S. 162
/12/ Interfaceplatinen für Breit- und Carrierband. Elektronik (1986) 8, S. 188
/13/ Kits offer MAP application development solution. Computer Design (1987) 15, S. 147
/14/ LAN controller converts PC to industrial workstation. Computer Design (1986) 22, S. 76
/15/ MAP-Controller für verschiedene Bus-Systeme. Elektronik (1987) 12, S. 50
/16/ Hard- und Software nach dem ISO/OSI-Standard. Elektronik (1985) 21, S. 42
/17/ Münzer, B.-G., Stachowiak, T.: 16-Bit-Signle-Board-Computer SBC 8086. Mikroprozessortechnik 1 (1987) 7, S. 200
/18/ Serielle Ein-/Ausgabesteuerung U 82530 DC/U 8030 DC. VEB Mikroelektronik „Karl Marx“ Erfurt, 1987

✉ KONTAKT ☎

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Wissenschaftsbereich Computertechnik, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 4 53 66

Innovative Computerarchitektur – Transputer

Dr. Jürgen Schlechter,
Salaheddin Dabbagh
Technische Universität Dresden, Sektion Informationstechnik

Zielstellung des folgenden Kurzbeitrages ist es, die innovative Architektur Transputer an Hand von gegenwärtig international angebotenen Schaltkreisen der Transputer-Familie, der Hochsprache OCCAM und der potentiell vorhandenen Anwendungsmöglichkeiten in der Telekommunikationstechnik vorzustellen.

Schaltkreise der Transputer-Familie

Der Begriff *Transputer*, vor etwa drei Jahren von der britischen Firma Inmos kreiert, hat sich inzwischen auf dem Gebiet innovativer Computerarchitekturen etabliert. Die Wortschöpfung *Transputer* soll dabei auf die Universalität wie beim Transistor und die Komplexität des Computers, von dem auch die Wortwendung herrührt, verweisen.

Der Transputer ist in seiner vervielfachten Nutzung speziell zur Realisierung arrayförmiger Strukturen, die vor allem Parallelverarbeitung ermöglichen, entwickelt worden. Er repräsentiert den allgemeinen Trend: „Weg vom Bus – hin zur Struktur“. Die folgende Darstellung (Bild 1) zeigt dabei mögliche Gedankengänge der Entwickler.

Der gegenwärtig leistungsfähigste Transputer ist der IMS T800, der als Weiterentwicklung des IMS T424 vor allem über eine zusätzliche Gleitkommaarithmetikeinheit verfügt.

Die weiteren Transputer-Schalkreise sind hauptsächlich auf die umfassende Nutzung des Link-Konzeptes ausgerichtet. So gewährleisten die Link-Adapter-Schaltkreise IMS C011 und IMS C012 eine Umsetzung von bytebreiten parallelen Schnittstellen in das serielle Link-Interface des Transputers. Das Transputer-Link-Protokoll ist dabei eine äußerst einfach gehaltene Vereinbarung über die serielle Datenübertragung in Form einer Punkt-zu-Punkt-Verbindung ohne direkt implementierte Fehlererkennungsmaßnahmen. Es wird über die Anschlüsse LinkOut (serieller Ausgabekanal) und LinkIn (serieller Eingabekanal) der Transputer-Schaltkreise abgewickelt (siehe Bild 2).

Dabei ist es durchaus möglich, über beide Kanäle eine Vollduplex-Übertragung durchzuführen, wobei in dieser Betriebsart die Nettoübertragungsrate selbst im Idealfall (Ein- und Ausgabeprozesse sind sofort aktionsfähig) für jeden Kanal dann auf maximal 61,53 % der Bruttoangaben zurückgeht.

Der Link-Switch-Schaltkreis IMS C004, der in einem 84poligen Pin-Grid-Gehäuse untergebracht ist, gestattet, 32 Link-Eingänge über einen Kreuzschienenverteiler (Crossbar-Matrix) auf 32 Link-Ausgänge durchzuschalten. Die entsprechende Zuordnung wird über ein spezielles Konfigurationslink (ConfigLinkIn/ConfigLinkOut) eingestellt und abgefragt. Mit diesem Schaltkreis ergeben sich weitere Möglichkeiten, auch dynamisch veränderbare Transputerstrukturen zu realisieren.

Der Transputer und OCCAM

Der Transputer und die Hochsprache OCCAM sind zusammen entworfen worden und bilden deshalb eine Einheit. Das OCCAM-Konzept hat zu einer im Verhältnis zu anderen Hochsprachen einfach strukturierten und überschaubaren Programmiersprache geführt, die insbesondere für Echtzeit- und Parallelverarbeitung Vorteile bietet. Die Idee des OCCAM-Programmiermodells besteht dabei in folgenden Grundsätzen:

In OCCAM werden Prozesse in Form von konkurrenten Systemen verbunden. Jeder Prozeß kann als „Black Box“ mit internen Zuständen dargestellt werden, wobei die Prozesse über die Punkt-zu-Punkt-Verbindung der Links miteinander kommunizieren können. Die Prozesse können genutzt werden, um eine Vielzahl unterschiedlicher Tatbestände (von der Logik-Nachbildung bis zur Verwaltungs-Aufgabe) zu repräsentieren.

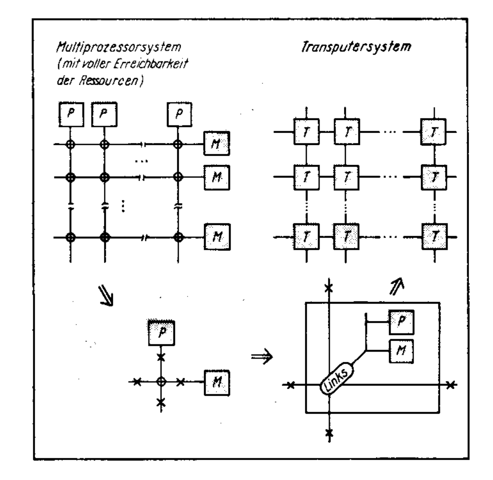
Die Prozesse sind in sich selbst geschlossen. Jeder Prozeß startet, realisiert eine bestimmte Anzahl von Aktionen und schließt dann. Eine Aktion kann formuliert werden durch: das Setzen von sequentiellen Abläufen (Prozessen), einer nach dem anderen wie in konventionellen Programmiersprachen, oder auch, und das ist sehr wesentlich, als eine Anzahl von parallelen Abläufen (Prozessen), die gleichzeitig ausgeführt werden können.

Ein Prozeß kann also selbst wieder aus einer Anzahl von Prozessen bestehen. Allgemein wird ein Prozeß aus den folgenden drei Primitiven konstruiert: Ausführung, Eingabe, Ausgabe. Ein in dieser Sprache implementiertes Schlüsselkonzept besteht darin, daß die Kommunikation synchronisiert und ungepuffert abläuft. Die Kommunikation läuft ab, wenn beide an der Kommunikation beteiligten Prozesse (der Eingabe- und der Ausgabeprozeß) bereit sind.

Da ein Prozeß eine interne Gleichzeitigkeit aufweisen kann, ist durch die Vielzahl der E/A-Links die gleichzeitige Kommunikation möglich. Als ein Wesensmerkmal des im Transputer implementierten OCCAM-Konzepts kann deshalb das Prinzip von Gleichzeitigkeit und Kommunikation angesehen werden. Das OCCAM-Prozeßmodell läßt sich dabei auf die unterschiedlich ausgebauten Hardware-Transputer-Systeme umsetzen. Die Konfiguierung des Systems (unter anderem welche Prozesse mit welchen Kommunikationsbeziehungen auf welchen Transputern laufen sollen) muß allerdings über ein Entwicklungssystem generiert werden.

So kann das gleiche OCCAM-Programm auf einer unterschiedlichen Anzahl von Transputer-Konfigurationen implementiert werden. Eine Konfiguration z. B. optimiert in bezug

Bild 1 Grundidee des Transputers (P – Prozessor, M – Memory, T – Transputer)



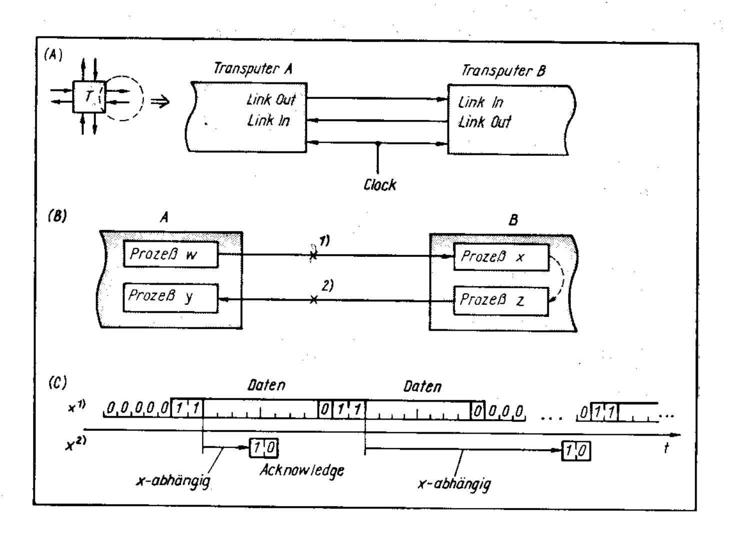


Bild 2 Serielle Kommunikation über die Transputer-Links
(A) physikalische Ebene
(B) Occam – Prozeßmodell der Punkt-zu-Punkt-Kommunikation
(C) Link-Protokoll

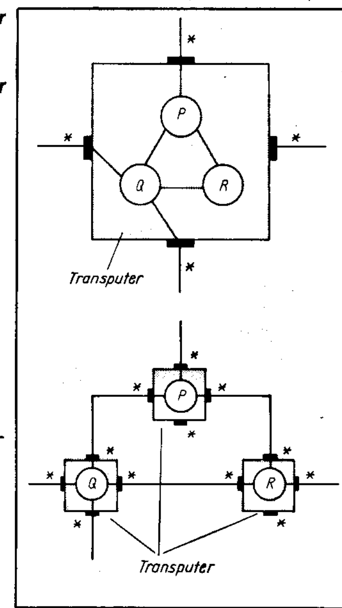


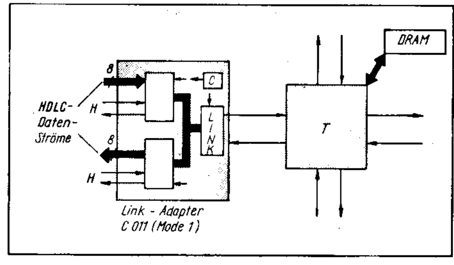
Bild 3 Prozeßmapping mit Transputer(n) (∗-Links)

auf die Kosten, eine andere auf Leistungsfähigkeit, oder wiederum andere stellen einen Kompromiß zwischen Kosten und Leistungsfähigkeit dar. Zur Verdeutlichung dieses Prinzips soll in Bild 3 das Prozeßmapping in einem oder verschiedenen Transputern aufgezeigt werden.
Wie bereits erwähnt, bietet das in OCCAM enthaltene Prozeßkonzept bei der Parallelverarbeitung große Vorteile. Es muß jedoch betont werden, daß die Parallelität nach wie vor vom Entwickler selbst projektiert und programmiert werden muß. Das Programmieren in OCCAM stellt in diesem Zusammenhang insbesondere folgende Anforderungen an den Programmierer:
– Erkennen der Prozesse, die parallel ablaufen können
– Zuweisung der Kanäle zur Kommunikation
– Vermeidung der Gefahr von Deadlocks
– Die Prozesse sollten gleichlang sein.
– Nutzung von Datenflußdiagrammen, da diese hierarchisch strukturiert sind.
OCCAM, als reine Parallelsprache für den Transputer angesehen, schöpft allerdings auch nicht alle Möglichkeiten dieser Architektur aus. So fehlt zum Beispiel bis auf die Array-Verarbeitung vor allem die Behandlung von Datenstrukturen, so daß damit Merkmale einer modernen Programmiersprache, wie eine rekursive Programmierung, nicht gewährleistet werden können. Deshalb und auch aus Gründen der bisherigen Nutzungsbreite werden zunehmend traditionelle Programmiersprachen (z. B. C) von der Host-Umgebung für den Transputer unterstützt.

Einsatzmöglichkeiten in der Telekommunikationstechnik

Der leistungsstarke Transputer mit seiner intern realisierten 32-Bit-CPU, die als RISC-Architektur (RISC–Reduced Instruction Set Computer; Rechner mit reduziertem, aber dafür rechenzeiteffektivem Befehlsschlüssel) konzipiert ist, gestattet günstige Nutzungsmöglichkeiten in der Echtzeitverarbeitung. Der wesentliche Vorteil des Transputers besteht dabei in der effektiven Bearbeitung von Prozessen mit echt flächenhafter oder räumlicher Parallelisierung. Außerdem besitzt er eine interne Struktur, die es gestattet, außerordentlich kommunikationsfreundliche, flächendeckende Computerstrukturen aufzubauen. Aus diesem Grunde wird er gegenwärtig vorrangig in der Bildverarbeitung (Image Processing) und als Koprozessor zur Leistungssteigerung appliziert. Obwohl er nicht primär für die Telekommunikationstechnik entwickelt wurde, bieten sich aber mit seinem kommunikationsfreundlichen Link-Konzept durchaus reale Anwendungschancen in diesem Fachgebiet an.
In /1/ wurde als eine Anwendungsmöglichkeit die X.25-Implementierung mit Transputer vorgestellt. Im vorliegenden Beitrag sollen Eignungsbetrachtungen für den Prozeß der Paketkonzentration dargelegt werden. Ausgangspunkt für diese Zielstellung ist es, obwohl durchaus die Nutzungsmöglichkeit telekommunikationstechnischer Spezialschaltkreise bekannt ist, ausschließlich Komponenten der Transputer-Schaltkreisfamilie zu applizieren. Dabei sollen an erster Stelle die Zugänglichkeit zum telekommunikationstechnischen Prozeß sowie entsprechende Leistungsabschätzungen vorgenommen werden.
Der Prozeß der Paketkonzentration bietet sich als experimenteller Ansatz auch aus Gründen einer hohen Adäquatheit von Prozeß und Architekturkonzept an. Dabei ist die optimale Version für diesen Applikationsfall eine hierarchische Baumstruktur, bei der die Transputer in den untergeordneten Zweigen vor allen Dingen die Ebene 2 – Verarbeitung übernehmen. In diesen und den nachgeschalteten Transputern kann weiterhin die Ebene 3 – Protokollverarbeitung erfolgen. Eine Basisaufgabe ist dabei die Behandlung von HDLC-Datenströmen. Dabei ergibt sich eine effektive Möglichkeit, die allerdings bereits an die Leistungsfähigkeit des Link-Konzepts sowie die Verarbeitungsleistung des Transputers heranreicht, durch die Nutzung des Link-Adapters IMS C 011 in Mode 1 für die Behandlung von acht HDLC-Kanälen (in Hin- und Rückrichtung) (Bild 4).
Die Auswertung der einlaufenden Datenströme kann dabei:
– synchronisiert auf eine teilbare Taktfrequenz entsprechend den vorgegebenen Normungen
– unsynchronisiert unter Einhaltung des Abtasttheorems bei Gewährleistung eines festen Taktnormals
erfolgen.
Im Transputer, der dann eine Datenstruktur entsprechend der vorgenommenen Datenstrom-Behandlung zur Verarbeitung erhält, sind z. B. folgende Algorithmen zu implementieren:
– Reformierung und Formierung des Datenstromes
– CRC-Berechnung und -Prüfung
– Datennormierung bei unterschiedlichen Bitraten
– Auslösung und Realisierung von Wiederholungsanforderungen
– Rahmenidentifizierung; zielgerichtete Steuerung der Informationsströme bzw. -pakete
– Initialisierung des Gesamtsystems
– Fehlerbehandlung und Ersatzschaltungsrealisierung.
Damit ergibt sich ein beachtlichen Echtzeitanforderungen entsprechendes, prozeßtypisches Aufgabenspektrum, das Transputervarianten größer 10 MIPS (MIPS – Million Instructions per Second) bedarf, um eine Umsetzung zu gewährleisten.
Insgesamt kann eingeschätzt werden, daß der Transputer als Architekturlösung für Prozesse mit innewohnendem Parallelisierungspotential eine effiziente Computerstruktur darstellt.

Bild 4 Transputereinsatz in der Telekommunikationstechnik (C – Control [Steuerung], H – Handshaking-Signale, T – Transputer, DRAM – dynamischer Schreib-/Lesespeicher)



Literatur
/1/ Kropf, R.G.: Paralleles Rechnen mit Transputern. Bulletin SEV/VSE (1988) 7/9, S. 356
/2/ Baron, J.; Cavill, P.; May, D.: Transputers does 5 or more MIPS even when not used in parallel. Electronics. November 17., 1983, S. 107
/3/ Bermond, R.: Transputer. Informatik-Spektrum (1986) 6, S. 359
/4/ Ebert, H.: Occamierung auf Transputern. Teil 1: c't (1988) 1, S. 138; Teil 2: c't (1988) 4, S. 204
/5/ Ebert, H.: Bei Fehlern einfach einfrieren. c't (1987) 12, S. 146
/6/ Ebert, H.: Starke Familienbande – die Transputerbausteine. c't (1987) 10
/7/ Ebert, H.: RISC schneller durch „Abspecken". c't (1986) 6, S. 40
/8/ Eckelmann, P.: Der Transputer als Grafikcontroller. Elektronik (1986) 24, S. 119
/9/ Eckelmann, P.: Neue Transputer leistungsfähiger. Elektronik (1986) 17, S. 22
/10/ Eckelmann, P.: Transputer – richtig eingesetzt. Elektronik (1985) 4, S. 57

Einsatz von Transputern in Spracherkennung und Robotik

Dietmar Neumerkel,
Prof. Dr. Dietrich Naunin
Technische Universität Berlin (West), Institut für Elektronik

In den letzten Jahren wurde in Großbritannien eine neue Prozessorfamilie entwickelt, die Transputer, deren leistungsfähigstes Mitglied ein 32-Bit-Prozessor mit Gleitkommarechenwerk ist. Besonders geeignet sind die Transputer für Mehrprozessoranwendungen. Das prädestiniert sie besonders für rechenintensive Aufgaben.
In dieser Vorstellung der Transputer soll zunächst das den Prozessoren zugrunde liegende Konzept verdeutlicht werden. Dann wird eine Leistungsabschätzung der Transputer im Vergleich zu anderen Mikroprozessoren angestellt. Im dritten Teil werden zwei Applikationen mit Transputern vorgestellt, die am Institut für Elektronik der Technischen Universität Berlin (West) Gegenstand zweier Forschungsschwerpunkte sind.

Transputer, eine neue Mikroprozessorgeneration

Das auffälligste Merkmal des Transputers gegenüber bisherigen Mikroprozessoren sind die zusätzlich zum Speicherinterface vorhandenen vier seriellen Schnittstellen, die sogenannten *Links* (Bild 1). Über diese schnellen, bidirektionalen Kanäle (20 MBit/s) können Punkt-zu-Punkt-Verbindungen zwischen den Transputern hergestellt werden, so daß eine Prozessorkopplung ohne Hardwareaufwand möglich ist. Dieses buslose Konzept ermöglicht eine mit steigender Prozessorzahl lineare Leistungszunahme des Gesamtsystems.
Im Bild 1 ist der 32-Bit-Prozessor *T800* im Blockbild dargestellt, wobei besonders dessen auf dem Chip integriertes Gleitkommarechenwerk auffällt.
Der T800 arbeitet mit einem internen Takt von 20 MHz; ein Speicherzugriff auf das interne, 4 KByte große RAM dauert nur 50 ns. Durch die RISC-Philosophie werden die meisten Befehle in einem Byte kodiert und in einem Zyklus bearbeitet.

Vergleich der Rechenleistung gegenüber anderen Prozessoren

Um bezüglich der Rechenleistung des Transputers zuverlässige Aussagen treffen zu können, wurden vergleichende Untersuchungen durchgeführt. Gegenstand dieses Benchmark-Tests war eine reellwertige 1024-Punkte-FFT nach Cooley-Tukey in 32-Bit-Gleitkommarechnung.

Die an dem Test beteiligten Signalprozessoren *TMS32020* (20 MHz) und *TMS320c25* (40 MHz) wurden in Assembler programmiert und verwendeten ein eigenes Zahlenformat. Die Prozessoren *MC68000* (8 MHz), *MC68020* (mit und ohne Arithmetik-Koprozessor *MC68881*, je 17 MHz) und *i80286* mit Koprozessor *i80287* (je 10 MHz) wurden in der Hochsprache **C** programmiert und verwendeten ebenso wie die Transputer ein IEEE-Zahlenformat. Die Transputer *T414* (wie T800, nur ohne Gleitkommaeinheit und mit nur 2 KByte RAM) und *T800* wurden in der speziell für die Transputer entwickelten Programmiersprache *OCCAM2* programmiert,

Tafel 1 Rechenzeit für die 32-Bit-FFT

TMS32020	TMS320c25	MC68000	MC68020	MC68020 MC68881	i80286 i80287	T414	T800
272 ms	141 ms	3749 ms	1178 ms	266 ms	803 ms	295 ms	55 ms

die auch die Formulierung von Programmen mit parallelen Prozessen erlaubt.
In der Tafel 1 und der Grafik in Bild 2 läßt sich sehr plastisch ablesen, wie leistungsfähig der Transputer auch in einer Einprozessorlösung ist. Es muß jedoch einschränkend hinzugefügt werden, daß sich bei anderen Anforderungen auch andere Testergebnisse ergeben werden. Zum Beispiel berechnet der *TMS32020* eine 1024-Punkte-FFT mit *16-Bit-Integer*-Operanden in 4 ms. Bei anderen Benchmark-Tests schließt die 2-Chip-Lö-

Bild 1 Blockbild des 32-Bit-Transputers T800

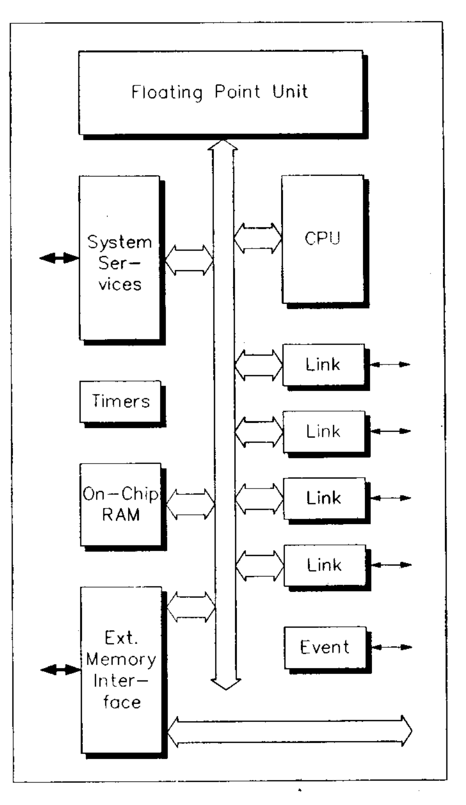
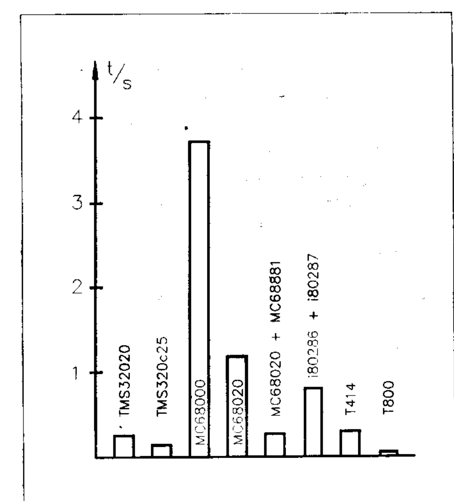


Bild 2 Rechenzeit für die 32-Bit-FFT



/11/ Eckelmann, P.: Transputer der 2. Generation. Elektronik (1987) 18–20
/12/ Electronics Newsletter: Transputer: Computer System Blazes at 3000 MIPS. Electronics (1986) 15, S. 79
/13/ Helzle, M.: Gar nicht Link! c't (1987) 11
/14/ Inmos: Transputer Reference Manual. Bristol. Okt. 1986
/15/ Klein, A.: Reduced Instruction Set Computer – Grundprinzipien einer neuen Prozessorarchitektur. Informatik-Spektrum (1986) 6, S. 334
/16/ Kropf, P. G.: A Comparision between the languages Chill and Occam. 4 th Chill Conference. Proc., München 1986
/17/ Transputer Module sind der neue Hit. Elektronik (1988) 4, S. 45
/18/ Nestyak, G.: Transputer. elektronik report (1988) 3, S. 45
/19/ Pountain, D.; Rudolph, R.: OCCAM – Das Handbuch. Verlag Heinz Heise GmbH, 1987
/20/ Sattelkau, M.: Transputer. Mikroprozessortechnik, Berlin 2 (1988) 5, S. 131

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Informationstechnik, Mommsenstr. 13, Dresden, 8027; Tel. 4 63 22 41

Computer-Club-Hinweis

Aufgrund eines notwendigen Beitragsaustausches erscheint der im Inhaltsverzeichnis des Heftes 12/1988 ausgewiesene Beitrag von Bernd Matzke: „SCP-Disketten-Directory als REDABAS-Datei" in Heft 2/1989, Seite 54.
Autor des ersten Beitrages „Mathematische Funktionen" einer Folge von REDABAS-Tips in MP 12/1988 ist Thomas Steffens.

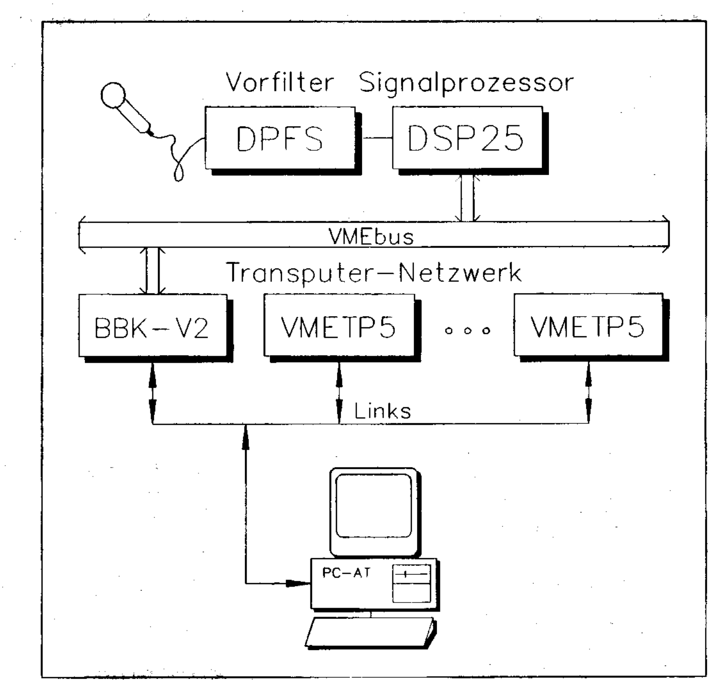
MP

sung *MC68020/MC68881* zum *T800* auf. Eine Leistungssteigerung durch den Mehrprozessoreinsatz ist jedoch nur mit den Transputern effizient möglich.

Blockschaltbild eines Spracherkennungssystems

Das Blockschaltbild eines Spracherkennungssystems mit Signalprozessor und Transputern wird in Bild 3 so vorgestellt, wie es am Institut für Elektronik der TU Berlin (West) konzipiert und realisiert wurde.
Da bei der Erzeugung der Merkmalsvektoren zur Spracherkennung *32-Bit-Integer*-Berechnungen ausreichen (Abtastwortbreite: 12 Bit), ist zur Signalaufnahme und -vorverarbeitung der Einsatz des Signalprozessors *TMS320c25* auf der Signalprozessorkarte *DSP25* einer Transputerlösung vorzuziehen. Von einem als Busmaster dienendem Transputer auf der *BBK-V2*-Karte (Firmenbezeichnung für Busbrückenkopf VME 2) werden die Merkmalsvektoren aus der *DSP 25* gelesen und der aus *T800* bestehenden Prozessorfarm auf den *VMETP5*-Boards über ein auf den *Links* basierendes Verbindungssystem zugeführt.
Die Vergleichsoperationen mit den hinterlegten Sprachmustern lassen sich ideal parallelisieren, so daß sich bei entsprechender Zahl von Transputern im System für jede Wortschatzgröße die angestrebte Reaktionszeit realisieren läßt.

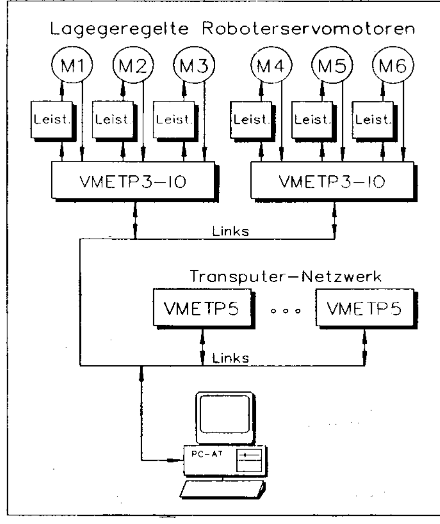
Bild 3 Blockbild eines Spracherkennungssystems



Einsatz von Transputern in der Robotik

Der Vorteil der hohen Rechenleistung eines Multi-Transputersystems soll auch in der Servoantriebstechnik bei hochdynamischen Antrieben und der Robotik genutzt werden (Bild 4). Für dieses Projekt wird eine modulare Transputerkarte *VMETP3-IO* zur Regelung von drei Motoren beliebiger Art (Gleichstrom-, Synchron- und Asynchronmotor) entwickelt, um damit die 6 möglichen Freiheitsgrade, die ein Roboter haben kann und die jeweils durch einen Motor verwirklicht werden, ausnutzen zu können. Über ein spezielles Verbindungssystem, das sich durch Software konfigurieren läßt, können beliebig viele Transputerkarten *VMETP5* zugeschaltet werden, um auch komplexe Regelungsalgorithmen, wie eine adaptive Regelung, zu implementieren.

Bild 4 Blockbild eines Robotersteuerungssystems



Zukünftige Einsatzmöglichkeiten

Durch die Entwicklung des Multi-User/Multi-Tasking/Multi-Processor-Betriebssystems *HELIOS* lassen sich neue Anwendungsfelder erschließen. So sind erstmals Mehrplatzrechner mit echter Parallelbearbeitung der einzelnen Benutzeraufgaben möglich.
Auch für andere, sehr aufwendige Rechenverfahren (Simulation, numerische Mathematik) stehen nun Systeme zur Verfügung, die ohne großen Hardwareaufwand die benötigte Verarbeitungsleistung bereitstellen können.

Wegbereiter der Informatik



BROOK TAYLOR

∗ 1685 Edmonton,
† 1731 London

Die überlieferten biographischen Angaben über den englischen Mathematiker Brook Taylor sind vergleichsweise spärlich. Immerhin ist bekannt, daß er ab 1701 in Cambridge vornehmlich Philosophie und Rechtswissenschaft und bei Isaak Newton Mathematik studiert hat. Beruflich war er ein vermögender Jurist und – ohne eine amtliche Stellung zu bekleiden – seit 1712 Mitglied der Royal Society; in den Jahren 1714–18 fungierte er auch als ihr wissenschaftlicher Sekretär. Er gehörte der Kommission an, die den Prioritätsstreit zwischen Leibniz und Newton untersuchte, und hier wirkte er in Briefen und Schriften verständlicherweise als eifriger Newtonianer.
In einem mathematischen Lexikon findet man mehrere Begriffe, die mit seinem Namen verbunden sind und noch heute zum fundamentalen Lehrstoff eines Mathematikstudiums gehören: Taylor-Interpolation, Taylor-Reihe, Taylorscher Satz, Taylorsche Quadraturformel. Sehr bedeutsam für die heutige Rechentechnik sind aus der Interpolationstheorie die Taylorschen Reihen geworden, also Potenzreihen der Form

$$f(x) = \sum_{k=0}^{\infty}(1/k!)f^{(k)}(x_0)(x - x_0)^k,$$

worin $f^{(k)}(x_0)$ die k-te Ableitung von f(x) an der Stelle x_0 bedeutet. Solche Reihen bilden bei jedem modernen Computer die Grundlage zur beliebig genauen Funktionswertberechnung bei allen möglichen transzendenten Funktionen, also z. B. bei der Logarithmusfunktion, der Exponentialfunktion, bei sin x bzw. cos x oder bei den hyperbolischen Funktionen. Die nach Taylor benannten Reihen kommen allerdings schon 1670 bei dem schottischen Mathematiker James Gregory (1638–75) sowie 1673 bei G. W. Leibniz vor. Taylor scheint davon nichts gewußt zu haben; in seinem Werk „Methodus incrementorum" (London 1715) leitet er die „Taylor-Reihe" aus Differenzenbetrachtungen her, und 1717 wendete er sie zur Gleichungsauflösung an. Taylor hat auch eine Näherungsformel für das bestimmte Integral

$$\int_a^b f(x)dx$$

angegeben, zu dessen Berechnung die ersten n Ableitungen von f(x) benötigt werden. Des weiteren untersuchte er das Problem der schwingenden Saite (1715) und hat 1717 ein schwieriges Leibnizsches Trajektorienproblem behandelt.

Dr. Klaus Biener

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
ISSN 0232-2892 · Mikroprozessortechnik · Berlin · 2 (1988) · S. 1–384

Jahresinhaltsverzeichnis 1988
VEB Verlag Technik Berlin

Mikroprozessortechnik

MP-Kurs

Informationen und Tips & Tricks

Berichte

Literatur

Börse

Entwicklungen und Tendenzen

Technik international/ vorgestellt

Autorenverzeichnis

Heft/Seite

Signalanalyse und Echtzeitverarbeitung mit digitalen Signalprozessoren

Dr. Jürgen Förster, Dieter Scholz
Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik

Einführung

Verfahren der digitalen Signalverarbeitung, insbesondere die Spektralanalyse auf der Basis der diskreten Fouriertransformation, finden in zunehmendem Maße Eingang in die Nachrichtentechnik, Schwingungsmeßtechnik, Sprach- und Bildverarbeitung. Neue Anwendungsgebiete stellen Maschinendiagnose, Prozeßüberwachung und Qualitätskontrolle dar.

Nach einer Periode, die vorwiegend durch an Großrechner gebundene Lösungen mit akademischem Charakter gekennzeichnet war, stehen dem Ingenieur seit den 80er Jahren leistungsfähige Spektralanalysatoren zur Verfügung, z. B. /1/. Trotz beachtlicher Leistungsmerkmale ist ihre Verbreitung durch zu geringe Flexibilität und einen relativ hohen Preis eingeschränkt.

Durch die breite Verfügbarkeit leistungsstarker Arbeitsplatzcomputer (PC) bis hin zu tragbaren Geräten einerseits und der Entwicklung von Prozessoren für Systeme mit hoher Verarbeitungsleistung andererseits (Arithmetikprozessoren, Signalprozessoren, Transputer) entstand die Voraussetzung für PC-gestützte Signalverarbeitungssysteme. In diesen PC-Meßgeräten wird der PC für die Menüführung, die Datenverwaltung und die grafische Darstellung verwendet. Zur Signalaufzeichnung und teilweise auch Signalverarbeitung dienen:

- Slotkarten für PC XT/AT (z. B. Signalprozessorslot von STAC /2/ mit NEC 77230, FFT mit N = 1024 in 9,4 ms)
- externe Module mit Buskopplung
- selbständige Module mit standardisierter Schnittstelle (PC-Instruments von Hewlett-Packard /3/ und Siemens /4/).

Für die weitere Signalverarbeitung im PC existieren international leistungsfähige Softwaresysteme wie ASYST, SIGNALYS und MATLAB.

Gegenstand des Artikels ist die Realisierung der Echtzeitspektralanalyse auf dem System TITAN. Dieses Signalverarbeitungssystem auf der Basis des Signalprozessors TMS 32010 erreicht im Echtzeitbetrieb einschließlich der Darstellung des Autoleistungsspektrums mit 512 Spektrallinien eine Abtastfrequenz von mehr als 30 kHz.

Vorgestellt werden weiterhin verschiedene Signalanalysestrukturen sowie ein Echtzeitrahmenprogramm.

Das Systemkonzept TITAN

Eine Analyse des internationalen Standes zeigt deutlich die Notwendigkeit von Applikationen der digitalen Signalverarbeitung im industriellen Bereich und deren fundierte Vorbereitung durch eine moderne Ausbildung unserer Studenten. Aufbauend auf langjährige theoretische Vorarbeiten entstand das Systemkonzept TITAN. Es besitzt folgende Einsatzcharakteristik:

1. Komplettes Entwicklungssystem für den Signalprozessor TMS 32010 mit Assembler und Debugger
2. Signalgenerator für verschiedenste im Host-PC generierte Signalfunktionen für Signalfrequenzen < 600 kHz
3. Signalanalyse mit verschiedenen Standardverfahren
4. Signalverarbeitung mit digitalen Verfahren und anschließende Digital-Analog-Wandlung

Das Gesamtkonzept ist in Bild 1 dargestellt. Es besteht aus einem Signalverarbeitungsmodul und einem Host-Rechner (PC-XT/AT, geplant ESER 1834/AC 7150). Die Verbindung erfolgt über eine parallele Schnittstelle.

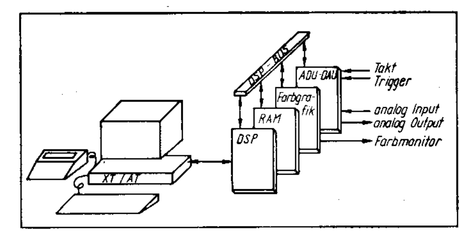


Bild 1 Gesamtkonzept TITAN

Die CPU-Karte basiert auf dem Signalprozessor TMS 32010 /5/ /6/. Dieser ist der erste Vertreter des abgestuften Familienkonzeptes digitaler Signalprozessoren der Firma Texas Instruments (Bild 2). Weitere herausragende Vertreter sind der TMS 320C25 /7/ mit einer Befehlszykluszeit von 100 ns und der TMS 320C30 /8/ mit 66 ns Zykluszeit und Gleitkommaarithmetik als Vertreter der 3. Generation. Wesentliche Leistungsmerkmale sind den angegebenen Literaturstellen bzw. Tafel 1 zu entnehmen.

Der unzureichend kleine interne Datenspeicher von 144 Worten wurde im System durch externe Speicherressourcen auf derzeitig 256 KWorte (maximal 1 MWort) erweitert. Für den Datenzugriff auf die Speicherressourcen dient eine äußerst effektive Pointerlogik, die neben der indirekten Adressierung auch parallele Adreßberechnung, zyklische Adreßzählung sowie das Bitreversing erlaubt.

Da gegenwärtig angebotene Grafikkarten keine schnelle grafische Darstellung zulassen, wurde eine Grafikkarte für 512 * 256 Punkte entwickelt, die zusätzlich eine Sonogrammdarstellung mit 64 Farben (geplant 256) ermöglicht.

Als Prozeßperipherie (ADU, DAU) wird gegenwärtig eine einkanalige Versuchsversion mit maximal 100 kHz Abtastfrequenz (12 Bit) eingesetzt. Geplant ist ein abgestuftes Spektrum zwischen mehrkanaligen *Low-cost*-Versionen (10 Bit, 30 kHz Abtastfrequenz) bis zu *High-speed*-Versionen (12 Bit, 1 MHz) für verschiedene Anwendungsfälle. Weitere Zielgrößen sind:

- Eingangsverstärker mit fester Verstärkung
- interner Takt in 500-ns-Schritten einstellbar
- externer Takteingang
- externer Triggereingang.

Damit steht ein flexibles System zur Verfügung, welches neben der Signalanalyse vor allem zum Entwurf und zur Realisierung digitaler Filter sowie zur Systemanalyse einsetzbar ist.

Die Realisierung der Diskreten Fouriertransformation (DFT)

Die Berechnung der DFT:

$$X(k) = 1/N \sum_{n=0}^{N-1} x(n) \cdot \exp(-j2\pi kn/N)$$

für ein Signal x(n) stellt eine Grundaufgabe der digitalen Signalverarbeitung dar. Sie wird mit Algorithmen zur schnellen Fouriertransformation (FFT) ausgeführt /9/.

Tafel 3 vergleicht die erzielten Transformationszeiten zweier FFT-Programme auf dem System TITAN mit anderen bekannten Lö-

Tafel 1 Leistungsmerkmale des TMS 32010

Wortbreite: 16 Bit
Befehlszykluszeit: 200 ns
Interner RAM: 144 Worte
externer Programmspeicher: 4 KWorte
16 × 16 Bit Multiplizierer
Barrel Shifter vor ALU
2 Auxilary Register für indirekte Adressierung
4 × 12 Bit Stack
Modifizierte Harvard-Architektur mit Programm- und Datenbus
16 I/O-Adressen
1 Interrupteingang

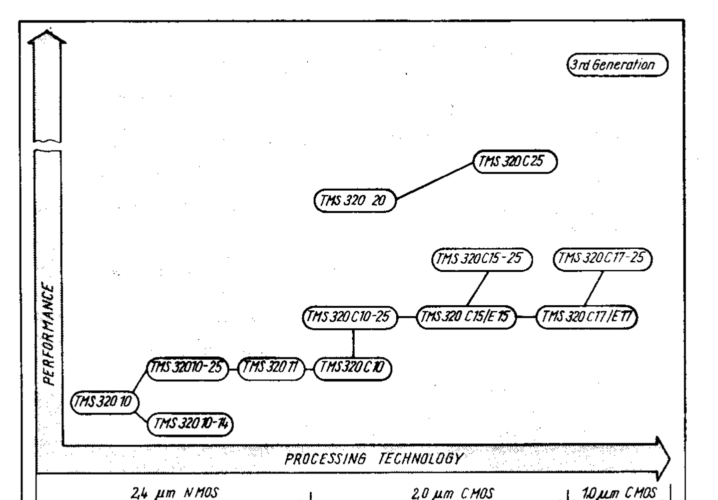


Bild 2 TMS320-Familienkonzept der Firma Texas Instruments (aus /8/)

Tafel 2 Abtastfrequenzen für Berechnung des Autoleistungsspektrums im Echtzeitbetrieb mit der Struktur IR

N	*Fs/kHz*
64	14,5
128	22,0
256	28,5
512	32,5
1024	34,5

sungen. Dabei können die Zeiten für andere TMS 32010-Systeme deutlich unterboten werden. Das Leistungsniveau des TMS 32020 ist erreichbar.
Diese Module stellen den Grundbaustein für ein Programm zur Berechnung des Autoleistungsspektrums (TSXX) dar.

Signalverarbeitungsstrukturen

Algorithmen zur Ermittlung von Signalkennfunktionen können in mehrere Signalverarbeitungsstrukturen eingebunden werden. Das soll anhand des Autoleistungsspektrums gezeigt werden, wobei von folgenden Aspekten ausgegangen wird:
– Unterscheidung zwischen *In-process-* und *Post-process*-Signalverarbeitung
– Unterscheidung zwischen *Real-time-* und *Intermediate*-Betrieb
– Einbeziehung von Mittelungsoperationen über mehrere Zeitsignale und Spektren als grundlegende Operation zur Reduktion irrelevanter Signalanteile.

Damit ergeben sich folgende Signalverarbeitungsstrukturen:

1. *In process real time* (IR; Bild 3a)
Das Aufzeichnen der Signale und die Verarbeitung von Signalausschnitten bzw. die Ergebnisauswertung erfolgen parallel, also in Echtzeit. Die Signalausschnitte können sich überlappen, müssen aber mindestens aneinanderliegend sein. Eine Mittellung über Signalausschnitte ist zwecklos, da beim Warten auf die Erfüllung von Triggerbedingungen ein Echtzeitfehler auftreten würde. Jedoch ist die Einführung einer Triggerbedingung für den Beginn der Analyse möglich. Eine Mittelung über mehrere Autoleistungsspektren sollte einbezogen werden. Diese Signalverarbeitungsstruktur ist zur Beobachtung instationärer Signale im Frequenzbereich geeignet.
2. *In process intermediate* (II; Bild 3b)
Das Aufzeichnen von Signalausschnitten und deren Verarbeitung erfolgen sequentiell. Die Einführung von Triggerbedingungen ermöglicht die Mittelung mehrerer Signalausschnitte. Auch die Mittelung über mehrere Autoleistungsspektren ist sinnvoll. Da gegenüber der IR-Struktur die Verarbeitungszeit der Signalausschnitte keine Rolle spielt, kann mit wesentlich höheren Abtastfrequenzen gearbeitet werden.
3. *Post process real time* (PR; Bild 3c)
Zunächst erfolgt die Aufzeichnung eines im Vergleich zu den Signalausschnitten für die Verarbeitung größeren Signalabschnittes, wobei Triggerung und damit die Mittelung über mehrere Signalabschnitte möglich sind. Bei der Verarbeitung einzelner Signalausschnitte können sich diese auch überlappen. Das entspricht dem *Real-time*-Betrieb. Für eine nachfolgende Verarbeitung ist die Speicherung aller Autoleistungsspektren sinnvoll (Wasserfalldiagramm, Sonogramm). Mit dieser Struktur ist die Beobachtung hochfrequenter instationärer Signalverläufe möglich.
4. *Post process intermediate* (PI)
Dieser Struktur liegt der Gedanke zugrunde, bei der PR-Struktur aus dem Signalabschnitt sich nicht überlappende Signalausschnitte zu verarbeiten, die über Triggerbedingungen ausgewählt und ebenfalls gemittelt werden können. Die Realisierung dieser Struktur erscheint nicht sinnvoll, da bei endlicher Länge des Signalabschnittes die Erfüllung einer vorgegebenen Anzahl von Triggerbedingungen

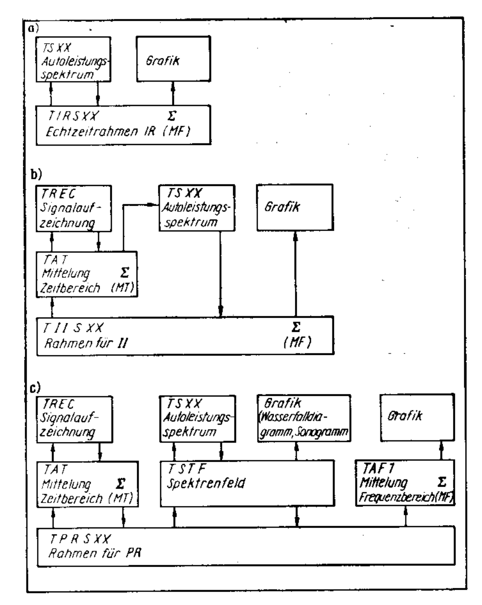


Bild 3 Signalverarbeitungsstrukturen
a) In process real time
b) In rocess intermediate
c) Post process real time

fragwürdig ist und damit Widersprüche innerhalb der Mittelungsaufgabe entstehen.
Im Konzept (Bild 3) sind nach den Routinen TREC und TSXX weitere Funktionsblöcke wie Signalfilterung, Filterung der Spektren und grafische Ausgabe der Signalausschnitte vorgesehen. Grundsätzlich sind diese Signalverarbeitungsstrukturen auch für andere Kennfunktionen wie Korrelationsfunktion, Amplitudenhistogramm und Kreuzleistungsspektrum sowie auf Identifikationsalgorithmen bei Klassifikationsaufgaben anwendbar.

Tafel 3 Rechenzeiten für FFT auf ausgewählten Signalprozessoren in ms

System	N 64/	128	256	512/	1024	2048	4096	Bemerkungen
TMS 32010/10/	2,87 1,92 1,72				69,4 45,3 42,3			Basis-2, 1 BF Basis-4, 1 BF Basis-4, 3 BF
TMS 32010/11/					59			Basis-2, 1 BF, + Window-Operation
TMS 32010 TITAN: FFT FFT1	0,9 1,0	2,35 2,4	5,9 6,1	13,8 14,5	32,4 34,0	77,8	176	
TMS 32020/12/ TMS 32020/13/		4,37	6,9 8,48 4,52		31.82			Basis 2 Basis 2, direkt programmiert

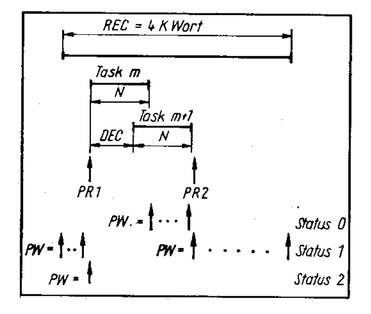


Bild 4 Zyklische Signalaufzeichnung und Aufruf einer Task

Realisierung eines Echtzeitrahmenprogramms

Da die Realisierung der Struktur IR die größten Anforderungen an Hard- und Software stellt, wurde als erstes ein Echtzeitrahmenprogramm entwickelt, welchem als Task das Autoleistungsspektrum, aber auch andere Signalkennfunktionen, zugeordnet werden können.
Anstatt des weitverbreiteten Einsatzes eines Eingangswechselspeichers, der die Verarbeitung aneinanderliegender Signalausschnitte gestattet, wird mit einem zyklischen Eingangsdatenspeicher der Länge REC = 4 KWorte gearbeitet. Auf den Eingangsdatenspeicher (Bild 4) greifen folgende 3 Pointer modulo REC zyklisch zu:
PW Adresse zum Einschreiben des nächsten Abtastwertes durch Interruptprogramme nach Ablauf der Abtastperiode Ta. Der Pointer wird nach jedem Schreibvorgang um 1 erhöht.
PR1 Anfangsadresse des der Task zugewiesenen Signalausschnittes
PR2 erste Adresse nach Ende des Signalausschnittes, welcher der nächsten Task zugewiesen wird.

Während der Bearbeitung einer Task können durch die Stellung des Pointers PW folgende 3 Statuszustände eintreten:

Status 0: PW befindet sich vor PR2. Ist Task m beendet, muß auf Status 1 gewartet werden, um Task m + 1 abzuarbeiten.
Status 1: Die notwendigen Abtastwerte für Task m + 1 sind vorhanden.

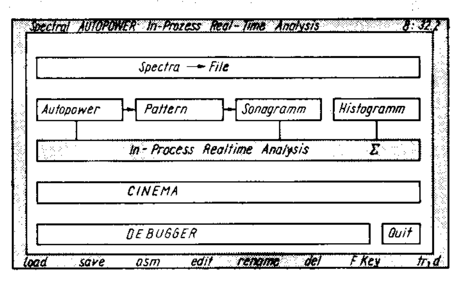


Bild 5 Steuermenü

Status 2: Die Verarbeitung von N Abtastwerten dauert länger als die Aufzeichnung von DEC Abtastwerten. Ausgabe eines Echtzeitfehlers. Die Signalaufzeichnung wird bis zur Abarbeitung von Task m beendet.

Wird die nächste Task aufgerufen, sind 6 ver-

schiedene Kombinationen der Pointer möglich, die innerhalb des Echtzeitrahmenprogramms ausgewertet werden müssen.
Somit können beliebig sich überlappende Zeitsignalausschnitte verarbeitet werden, wenn die Bedingungen:

REC > N + DEC + 2 und **N > 1**

erfüllt sind. Das ermöglicht Overlapp-add-Entwicklungen für verschiedene Fensterfunktionen /14/. Weiterhin realisiert das Echtzeitrahmenprogramm den Aufruf von Vorprogrammen sowie die Mittelung mehrerer Signalkennfunktionen. Bei der Anwendung auf die Berechnung des Autoleistungsspektrums im Echtzeitbetrieb mit Fensterfunktion, Sonogrammdarstellung und ohne Mittelungen ergeben sich für DEC = N die in Tafel 2 dargestellten maximalen Abtastfrequenzen.

Menügestaltung

Die Gestaltung der Bedienoberfläche ist ausschlaggebend für die Akzeptanz oder Ablehnung eines Programms durch den vorgesehenen Nutzerkreis. Bild 5 zeigt das Steuermenü für das oben vorgestellte Echtzeitrahmenprogramm zur Autospektralanlyse. Durch Betätigung der Kursortasten wird ein aktueller Funktionsblock ausgewählt und markiert. Nach Betätigen der ESCAPE Taste können in einem weiteren Menü die zugeordneten Parameter geändert werden. Mit der Taste ENTER wird die Abarbeitung der Blockfunktion gestartet, die durch einen weiteren Tastendruck unterbrechbar ist.
Zur Verdeutlichung des Signalflusses ist im zentralen Teil des Steuermenüs die Signalverarbeitungsstruktur IR dargestellt. An die Autoleistungsspektralanalyse schließt sich wahlweise eine Kurven- oder Sonogrammausgabe der Teilspektren an. Zugleich wird das dezentrale Konzept der Parametereingabe deutlich. Als erstes erfolgt die optimale Einstellung der einzelnen Funktionsblöcke, z. B. Autopower oder Histogramm, die auch einzeln gestartet werden können. Dann wird zur nächsten Hierarchiestufe übergegangen.
Mittels des Debuggers können während der ersten Nutzungsphase noch Veränderungen auf der TMS-Ebene vorgenommen werden. Im Funktionsblock *Spectra* --> *File* ist die Übergabe der Spektren auf Files vorgesehen, die mit kommerziellen Signalverarbeitungssystemen, z. B. MATLAB, kompatibel sind.

☒ KONTAKT ☎

TU Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 5 61 33 31

Literatur

/1/ Schmalband-FFT-Spektrumsanalysatoren 2032, 2033 und 2034. Bruel & Kjaer Katalog 1988, DK-2850 Naerum Dänemark, S. 50
/2/ Angebot der STAC Elektronische Systeme GmbH, c't Hannover (1988) 9, S. 215
/3/ Lightman, S.: PC-Meßgeräte. Elektronik (1985) 16, S. 52
/4/ Damit Meß- und Prüfgeschichten kürzer werden: die neuen PC-Meßgeräte. Messekatalog der Siemens AG auf der Leipziger Frühjahrsmesse 1988
/5/ Chance, J.:TMS 320 digital signal processor development system. microprocessors and microsystems, vol 9 no 2, march 1985
/6/ TMS 32010 USER'S GUIDE 1983
/7/ TMS 320C25 USER'S GUIDE 1986
/8/ Kun-Shan Lin; Frantz, G. A.; Simar, R.: The TMS 320 Family of Digital Signal Processors. Proceedings of the IEEE, Vol. 75, No. 9, September 1987, S. 1143
/9/ Förster, J.: Zur Realisierung der Diskreten Fourier Transformation. Mikroprozessortechnik, in Vorbereitung
/10/ Burrus, C. S.; Parks, T.: DFT/FFT and Convolution Algorithms. New York: Wiley 1984, S. 156
/11/ Kueng, R.: Flexibler FFT-Prozessor löst Echtzeitaufgaben. Elektronik (1986) 21, S. 101
/12/ Wiggins; Mcmahan; Tarrant; Gass: DSP boards help tackle a tough class of AI Tasks. Elektronics, August 21., (1986), S. 64
/13/ Papamichalis, P.; So, J.: Implementation of Fast Fourier Transform Algorithms with the TMS 32020. Digital Signal Processing Applications Volume 1, Prentice Hall and Texas Instruments 1987, S. 92
/14/ Hardtke, H.-J.; Le huy Thao: Fourieranalyse mit gleitendem Fenster (Kurzzeit-Fourier-Analyse). Messen Steuern Regeln, Berlin 29 (1986) 4, S. 186

Gate-Array und Standardzelle – dominierende Vertreter innerhalb der ASICs

Prof. Dr. Dietmar Müller
Technische Universität Karl-Marx-Stadt
Sektion Informationstechnik

Mit den in /1/ angegebenen Kurzcharakteristiken und den in /2/ beschriebenen Kriterien für ASICs ergibt sich Bild 1. Dabei ist die Zweiteilung in Anwenderprogrammierbare ASICs und Maskenprogrammierbare ASICs, die damit entwerfbar sind, zu erkennen.

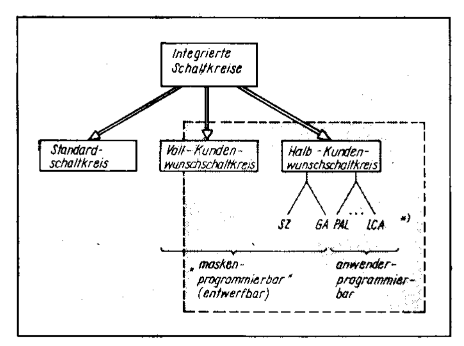


Bild 1 Einordnung von ASICs in die Schaltkreisarten
****) Erläuterung: siehe MP 11/88, Seite 324***

Anwenderprogrammierbare ASICs

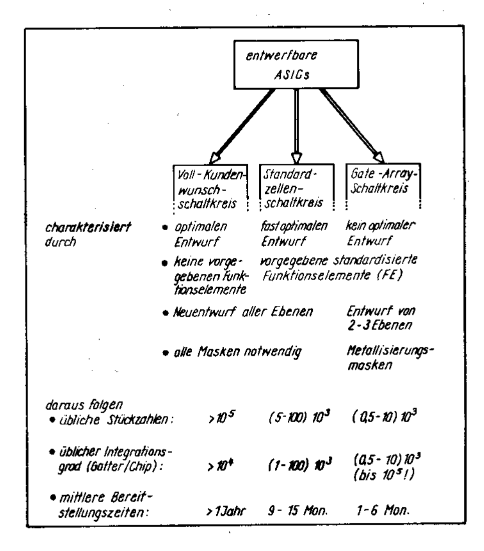
sind durch die schnelle Individualisierung, auch oft als Personalisierung bezeichnet, bei geringer Komplexität gekennzeichnet. Innerhalb dieser Gruppe nehmen die *LCA (Logic Cell Array)* eine Sonderstellung ein, weil diese gegenwärtig bereits 2000 bis 3000 nutzbare Gatteräquivalente (GE- siehe dazu /1/) aufweisen und nach /3/ bereits LCA-Schaltkreise mit 9000 GE angeboten werden. Wegen dieser Quantität und dem mit der Gate-Array-Architektur (Logik-Zellenfeld mit umgebende E/A-Zellenring) vergleichbaren Chipaufbau werden LCA oft auch als programmierbare Gate Array bezeichnet.
Die Anwenderprogrammierung der LCA erfolgt, indem von einem EPROM in den eingebauten LCA-Schaltkreis die Steuerinformationen für die Vielzahl der Schalttransistoren in entsprechende RAMs des LCA eingelesen wird. Diese Schalttransistoren verbinden die Logikblöcke des LCA (CLB = Configurable Logic Blocks) so, daß die mit Hilfe eines speziellen CAD-Systems entworfene und simulierte Schaltungsstruktur eingestellt wird.
Diese perspektivreiche ASIC-Art wird in einem der nachfolgenden Artikel der ASIC-Serie näher vorgestellt.

Maskenprogammierbare ASICs

werden besonders durch die Gate-Array- und die Standardzellen-Schaltkreise geprägt. Während bisher noch Unklarheiten über den jeweils größeren Zuwachs bei *Gate-Array* oder Standardzelle existierten, ist auf der ISSCC '88 (International Solid State Circuits Conference) /4/ eindeutig die „Gate-Array-Philosophie" als die dominierende bei den ASICs favorisiert worden. Gleiches trifft bei den Technologien für die CMOS-Technologie gegenüber bipolaren sowohl für die Speicher- als auch für die ASIC-Schaltkreise zu.

Die *Standardzellen*-Schaltkreise besitzen zwar ausgeprägte Möglichkeiten zur Entwurfsoptimierung und damit zur Verringerung der Chipfläche, was über die Erhöhung der Ausbeute zur Verringerung der Kosten führt; dennoch wird von zahlreichen Autoren der Standardzellen-Entwurf mehr als Rationalisierungsmittel zum Entwurf echter Kundenwunsch-Schaltkreise durch professionelle Entwerfer und weniger als ASIC-Entwurf durch potente Anwender betrachtet.
Die Gemeinsamkeiten und die Unterschiede der maskenprogrammierbaren ASICs sind in Bild 2 für den Entwurf und die Präparation (angedeutet durch die Maskenanzahl) schematisch dargestellt und sollen nachfolgend für die Gate-Array- und Standardzellen-Schaltkreise ausführlicher dargelegt werden, wobei die Kurzcharakteristik dieser ASIC-Vertreter in /1/ angegeben ist.

Bild 2 Charakteristika von (entwerfbaren) ASICs



Gemeinsamkeiten von Gate-Array- und Standardzellen-Schaltkreisen

basieren auf der ausschließlichen Nutzung von standardisierten Funktionselementen (Makros, Bibliothekselementen) beim Entwurf. Diese standardisierten Funktionselemente, die bei Standardzellen-Schaltkreisen eben Standardzellen genannt werden und die damit den unscharfen Begriff „Standardzellen-Schaltkreise" geprägt haben, sind vollständig entworfen, das heißt dimensioniert und das elektrische Verhalten simuliert und mittels Teststrukturen auch verifiziert. Die Funktion und die Parameter der Funktionselemente sind für den Kunden damit festgelegt. Er kann diese wie die Standard-SSI-Schaltkreise und MSI-Schaltkreise der bekannten TTL- und CMOS-Reihen nutzen, deren konkrete Funktionen und Parameter ja auch fixiert sind. Dieser (scheinbare) Nachteil für den Schaltkreisentwerfer hat jedoch den großen Vorteil, daß er im Rahmen der funktionellen Möglichkeiten dieser Elemente sich auf den eigentlichen Schaltungsentwurf konzentrieren kann und nicht noch Transistorstrukturen entwerfen und deren Parameter fixieren muß. Die sehr aufwendige elektrische Dimensionierung wird bei einmaligem Aufwand vom Hersteller oder vom Anbieter des Entwurfs- und Technologiesystems durchgeführt.

Unterschiede von Gate-Array- und Standardzellen-Schaltkreisen

resultieren aus der Anwendung dieser standardisierten Funktionselemente beim Entwurf und der Herstellung.

Beim Gate-Array-Schaltkreis sind die Anzahl und die Lage der Funktionselemente ebenso wie die Chipfläche, die Anschlußzahl, die Anzahl, die Lage und die Aufnahmefähigkeit der Verdrahtungskanäle vom Hersteller fixiert. Der Entwerfer bzw. der Anwender kann nur im Rahmen dieser Vorgaben „seinen" Gate-Array-Schaltkreis entwerfen. Daraus resultiert aber der gravierende Vorteil, daß wegen dieser fixierten Vorgaben die notwendigen Transistorstrukturen vollständig auf den Siliziumscheiben (master slice) vorgefertigt werden können. Diese auf Vorrat produzierten Scheiben werden später durch die Verbindung der vorgefertigten Strukturen dem speziellen Anwendungsfall angepaßt, wozu nur noch 20–30 % des Gesamtaufwandes notwendig sind.

Beim Standardzellen-Schaltkreis kann der Entwerfer bzw. der Anwender die Anzahl und die Lage der Funktionselemente – entspricht den Standardzellen – im Rahmen der technologischen Möglichkeiten frei wählen. Gleiches gilt damit auch für die Chipfläche und die Pinanzahl, die im Rahmen verfügbarer Bondinselringe wählbar sind. Aus diesem Vorteil, einem freizügigen Entwurf, folgt der Nachteil – zumindest eine zeitliche Verlängerung der Bereitstellung –, daß alle Ebenen und damit alle Masken für jeden Einsatzfall neu entworfen werden müssen. Während der manuelle Entwurfsaufwand durch Nutzung leistungsfähiger CAD-Systeme vergleichbar wird mit dem minimalen Aufwand bei Gate-Array-Schaltkreisen, resultiert aus der erst während des Entwurfs fixierten Lage der Transistoren die Unmöglichkeit einer Vorfertigung, wie bei Gate-Array-Schaltkreisen üblich. Dies bedeutet, daß für Standardzellen-Schaltkreise erst nach dem Entwurf die technologische Präparation begonnen werden

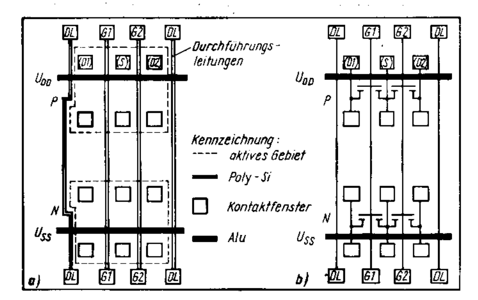


Bild 3 Layout und Transistorstruktur einer Halbzelle eines Gate-Array-Masters

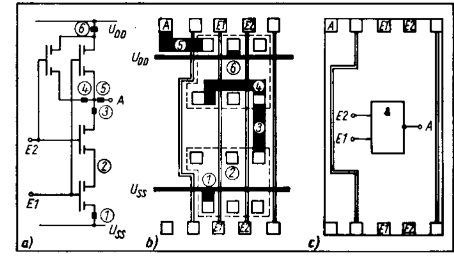


Bild 4 Verschiedene Darstellungsformen für das Funktionselement: 2fach-NAND

kann, so daß die Bereitstellungszeit bei Standardzellen-Schaltkreisen wesentlich länger als bei Gate-Array-Schaltkreisen ist. Diesem Nachteil stehen natürlich große Vorteile gegenüber. So sind z. B. eine optimale Größe der Chipfläche, so sind Funktionselemente mit angepaßt dimensionierten Transistoren (da keine universell dimensionierten Transistoren notwendig sind, kann Last- und/oder Zeitverhalten den verschiedenen Funktionselementen angepaßt werden), und so sind Bus-Strukturen möglich. Bemerkenswert ist, daß innerhalb der Standardzellen-Bibliothek auch analoge Funktionselemente, wie Digital-Analog- und Analog-Digital-Wandler möglich sind. Eine Gegebenheit, die beim Gate-Array-Entwurf erst als Tendenz vorhanden ist (BiCMOS).

Der große Vorteil der Gate-Array-Schaltkreise – die schnelle Verfügbarkeit von Musterschaltkreisen – gegenüber Standardzellen-Schaltkreisen wird durch eine weitere Art von Gate-Array, dem *Lasarray* (z. B. /5/) weiter ausgeprägt. Gleichzeitig gelingt es mit dieser Variante, die für Gate-Array-Schaltkreise optimale Stückzahl von 500 bis 10000 Stück (siehe Bild 2) weiter zu verringern. Beim Lasarray bewegen sich die nutzbaren Gatteräquivalente im unteren Bereich der Gate-Array-Schaltkreise (300 bis 3000), wobei die Chiparchitektur und die Entwurfsmethodik sich stark ähneln. Im Gegensatz zur Maskenprogrammierung des Gate-Array werden die Lasarrays durch einen Laserstrahl individualisiert (siehe oben). Durch die Laserstrahldirektbelichtung werden die Photoschichten so belichtet, daß durch nachfolgende Ätzprozesse die spezielle Schaltungsstruktur entsteht, die der Anwender mittels CAD-System entworfen hat. Diese Entwurfsdaten steuern den Laserstrahl im obigen Prozeß, wodurch CAD und CAM perfekt verbunden sind.

Es muß betont werden, daß diese Lasarrays keine Alternative zum Gate-Array sind, sondern eine sinnvolle Ergänzung für kleine Stückzahlen.

Zum prinzipiellen Verständinis des Gate-Array-Entwurfes auf der Basis standardisierter Funktionselemente sollen diese nachfolgend erläutert werden, obwohl deren Entwurf – wie oben beschrieben – *nicht* die Aufgabe des (Anwender-) Entwerfers ist und obwohl der Entwurf von Gate-Array-Schaltkreisen erst im nächsten Artikel der ASIC-Serie detailliert erläutert wird.

Funktionselemente für den Gate-Array-Entwurf

entstehen durch die Verbindung der auf dem Untergrund vorgefertigten Transistoren. Diese Transistorfelder sind gegenwärtig noch vorwiegend zeilenförmig angeordnet, wobei sich zwischen diesen Zeilen die Verdrahtungskanäle befinden (siehe Bild 5).

Da international und national die Bedeutung der CMOS-Technologie ständig wächst, beziehen sich die folgenden Erläuterungen des Layouts von Funktionselementen auf CMOS, zumal die DDR-Gate-Array-Systeme U 5200 /6/, U 5300 und auch das Nachfolgesystem auf der CMOS-Technologie in den entsprechenden Technologieniveaus beruhen.

Bild 3 zeigt das Layout und die zugehörige Transistorstruktur einer Halbzelle eines Gate-Array-Masters. [1]

Dieser Master stellt die Grundlage eines „CMOS-Gate-Array-Lehrsystems" /7/ dar. Aus Gründen der Übersichtlichkeit und des besseren Verständnisses weisen die Transistoren eine geradlinige Struktur auf, und es besitzen die p- und n-Transistoren gleiches Längen-/Breitenverhältnis, obwohl dies für die Verzögerungszeiten bei realen Strukturen ungünstig ist.

Auf der Grundlage einer oder mehrerer derartiger Halbzellen werden die erforderlichen Funktionselemente entworfen. Dies soll am Beispiel eines Zweifach-NAND erläutert werden. Hierzu ist im Bild 4a die notwendige Transistorschaltung, die in CMOS die Funktion eines zweifachen NAND realisiert, gezeichnet. Im Bild 4b ist auf der Basis der Transistorstruktur einer Halbzelle (siehe Bild 3) das Zellenlayout des zweifachen NAND dargestellt. Die stark markierten Stellen bezeichnen die „Verdrahtung", die in diesem Fall als Alu-Leitbahnen ausgeführt ist. (Die den stark markierten Stellen zugeordneten Ziffern sind zugefügt worden, um das Verständnis zu erhöhen.)

Dieses Zellenlayout ist für den Anwender uninteressant, er kann es nicht beeinflussen, ja, er muß und wird es in der Regel nicht kennen. Ihm werden neben der logischen Funktion die dynamischen Parameter und die Anschlußparameter zur Kenntnis gebracht, so daß er mit dem symbolischen Layout der Funktionselemente (Bild 4c) entwerfen kann. Im allgemeinen entwirft er sogar nur mit dem logischen Symbol der Funktionselemente oder deren sprachlicher Beschreibung.

Der Entwickler eines Gate-Array-Systems entwirft eine Vielzahl von Funktionselementen. Diese müssen unter Beachtung der Entwurfsregeln sehr genau mit rechentechnischen Mitteln und durch Herstellen von Teststrukturen untersucht werden, damit die Parameter auch unter Worst-case-Bedingungen eindeutig bekannt sind. Die rechentechnische Simulation des elektrischen Verhal-

[1] Bei farbigen Darstellungen von Layouts sind das aktive Gebiet *rot*, das Poly-Silizium *blau*, die Kontaktfenster *grün* und die Alu-Bahnen *schwarz* dargestellt.

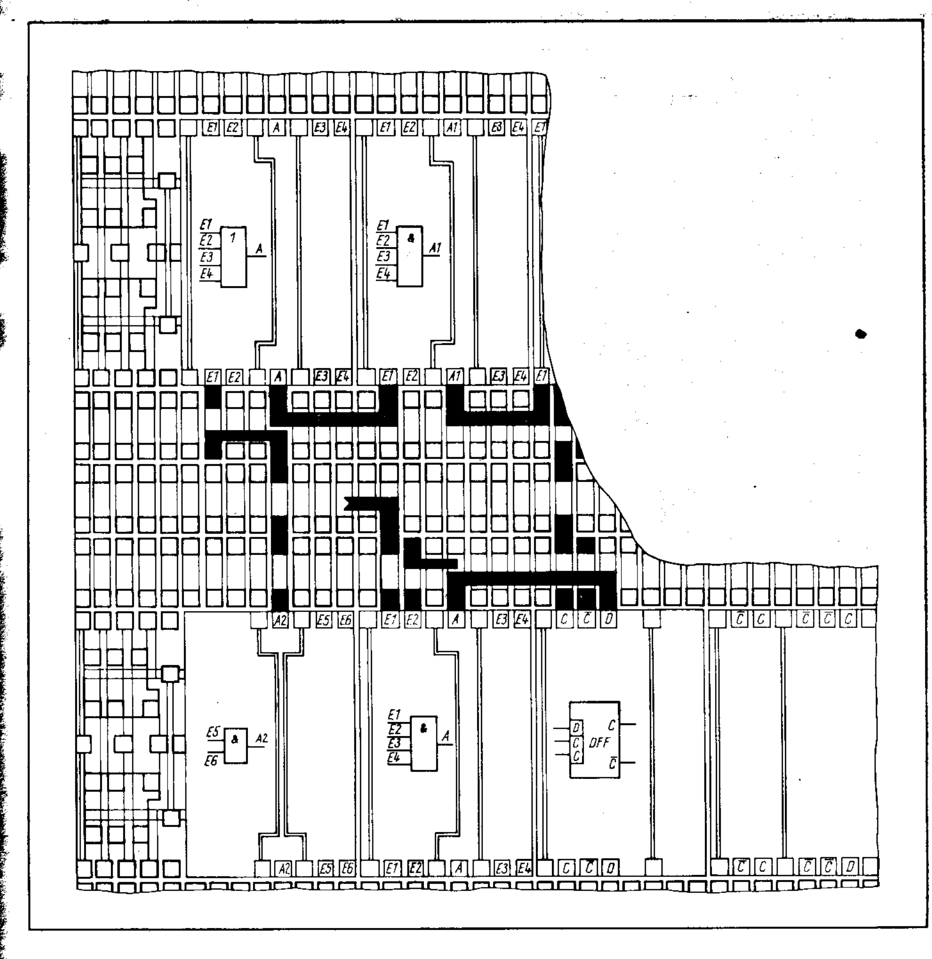


Bild 5 Ausschnitt eines Entwurfes mit einem „CMOS-Gate-Array-Lehrsystem" (teilweise trassiert)

tens mittels Netzwerkanalysemethoden ist daher sehr rechen- und zeitaufwendig. Sie muß jedoch nur beim Entwickler des Systems, quasi einmal, für alle nachfolgenden Anwendungen durchgeführt werden.
Diese Funktionselemente mit ihren Parametern stehen dem Anwender in einer Bibliothek von Funktionselementen zur Verfügung (im System U 5200 als Macrobibliothek /6/ bezeichnet). Die konkret nutzbaren Funktionselemente sind vom Entwurfssystem, vom Technologieniveau und von der Zahl der vom Anwender nutzbaren Ebenen (z.B. Alu – Poly-Si – Kontaktfenster oder Alu – 2mal Poly-Si – Kontaktfenster oder 2mal Alu – ...) abhängig.
Allen gemeinsam ist die Beschränkung auf vorerst digitale Grundelemente. Dies bedeutet eine Vielzahl von kombinatorischen Grund- und Komplexgattern und nur einen (oder wenig modifizierten) FF-Typ (siehe dazu nächste Artikel der ASIC-Serie). So sind z. B. möglich

- Inverter (einfach, mehrfach)
- zwei- bis vierfach NAND/AND
- zwei- bis vierfach NOR/OR
- zweifach Antivalenz/Äquivalenz
- Multiplexer, Demultiplexer
- Komplexgatter (Kombination von NAND-NOR-Strukturen)
- D-FF, J-K-FF und Modifikationen.

Die Bibliothek der Funktionselemente ist oben offen, das heißt, geforderte Funktionselemente können bei Sinnfälligkeit nach entsprechender Simulation vom Entwickler des Gate-Array-Systems aufgenommen werden. Mit dem konkreten Wissen um die Parameter dieser Funktionselemente – vergleichbar mit dem Wissen um die Parameter der TTL- und CMOS-Standardschaltkreise – wählt der Anwender (zumindest bei einem manuellen Entwurf, z. B. während der Aus- oder Weiterbildung) die für die Realisierung seiner erforderlichen Funktion notwendigen Funktionselemente aus (vergleichbar mit der Auswahl obiger Standard-Schaltkreise). Anschließend erfolgt ein „geschicktes" Anordnen (Plazieren) der ausgewählten Funktionselemente, damit die Verbindung der Funktionselemente vollständig möglich wird (Trassierung). Die Plazierung und Trassierung ist vergleichbar mit der Anordnung von Standard-Schaltkreisen beim Entwurf einer kreuzungsfreien Zwei-Lagen-Leiterplatte.
Ein Ausschnitt des Lehrmasters mit plazierten symbolischen Layouts und einer stückweisen Trassierung zeigt Bild 5. Es ist mit Hilfe der Erläuterung im Bild 3 zu erkennen, daß die Verdrahtungskanäle Alu-Leitbahnen (stark schwarz) aufweisen, die über die Poly-Silizium-Bahnen (dünn markiert) angeordnet werden. Beide Leitbahnen können durch die Kontaktfenster (Quadrate – mittelstark gezeichnet) verbunden werden.
Aus dem Bild 5 und der skizzierten Beschreibung des (manuellen) Entwurfs eines Gate-Array-Schaltkreises kann die große Ähnlichkeit zum Leiterplattenentwurf erkannt werden. Der skizzierte manuelle Entwurf ist selbstverständlich nicht erstrebenswert und bei der gegenwärtig bereits verfügbaren Zahl von Gatteräquivalenten (größer 1000) auch nicht realisierbar.
Wie in /2/ kurz erläutert, kann der Anwender „seinen" Gate-Array-Schaltkreis nur effektiv entwerfen, wenn neben der Technologie ein leistungsfähiges CAD-System bereitgestellt wird. Daher wird im nächsten Artikel der ASIC-Serie der rechnerunterstützte Entwurf von Gate-Array-Schaltkreisen erläutert und die Nutzung der Systeme ARCHIMEDES des VEB Forschungszentrum Mikroelektronik Dresden (ZMD) und PC-GAD der Technischen Universität Karl-Marx-Stadt beschrieben.

Literatur

/1/ Müller, D.: ASIC von A bis Z (C). Mikroprozessortechnik, Berlin 2 (1988) 11, S. 324
/2/ Müller, D.: ASIC – eine Revolution? Mikroprozessortechnik, Berlin 2 (1988) 11, S. 323
/3/ Logic Cell Arrays von XILINX. Elektronik (1987) Sonderheft 237, S. 73
/4/ Winkler, P.: International Solid State Circuits Conference 1988. Reisebericht, Dresden 3/1988
/5/ Soldani, L.: ASICs per Laserstrahl (Alternative für kleine Mengen – das Lasarray). Elektronik (1987) Sonderheft 237, S. 71
/6/ Fischer, W.-I. u. a.: CMOS-Gate-Array-System U 5200. Nachrichtentechnik, Elektronik, Berlin 36 (1986) 1, S. 21
/7/ Müller, D. u. a.: Entwurfspraktikum mit einem Gate-Array-Lehrsystem an der TH Karl-Marx-Stadt. Vortrag 1. Tagung Schaltkreisentwurf, Dresden 1986

✉ KONTAKT ☎

Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022; Tel. 5 61 31 95

TERMINE

Die Fachtagung „Aufsetzmontage in der Leiterplattentechnik unter Berücksichtigung der Nacktchipverarbeitung" mit internationaler Beteiligung aus den RGW-Ländern findet mit **neuem Termin**
vom 27. bis 28. September 1989
in Berlin statt.
Vortragsmeldungen sind bis zum 7. Februar 1989 zu richten an:

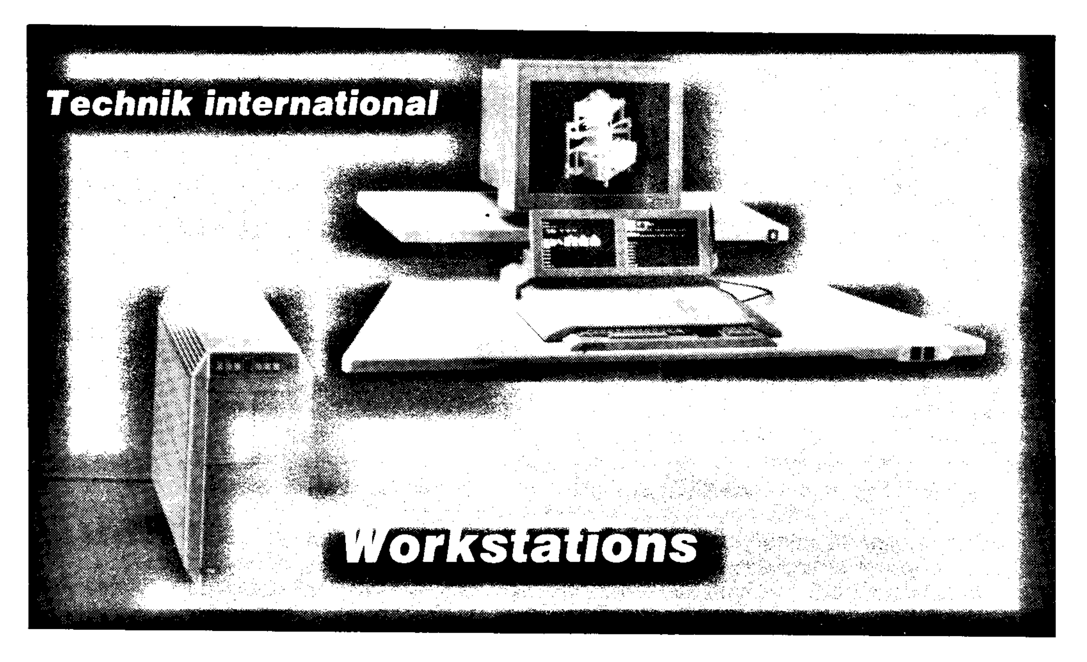
Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Postfach 1315, Berlin, 1036.

Hoppe

Leeranweisung

Zeichnung: Steger





ren Rechnerleistung bis zu 10 MIPS, der Hauptspeicher 8 bis 16 MByte und die Bildschirme bis zu 4096 × 4096 Pixel haben. Als Quasi-Industrie-Standard für die Kommunikation hat sich Ethernet mit 10 MBit/s herausgestellt. In dieser Klasse haben sich eindeutig die 32-Bit-Mikroprozessoren wie MC 68020, MC 68030 und Intel 80386 durchgesetzt, die jetzt zunehmend durch RISC-Prozessoren verdrängt werden.
Das Bild zeigt als Beispiel eine interessante Neuentwicklung des norwegischen Unternehmens Norsk Data auf der Basis eines Drei-Prozessorsystems. Der Grafikrechner beruht auf dem Mikroprozessor MC 68020 und einem AMD-Chip für die Pixel-Grafik. Damit bei Konstruktionsaufgaben keine Bildschirmflächenanteile für Bedien- und Kommunikationsaufgaben (Multiwindowing) verloren gehen, werden dafür gesonderte Bildschirme eingesetzt.

Der Begriff der Workstation wurde Anfang der achtziger Jahre von der amerikanischen Firma Apollo Domain geprägt. Man verstand darunter einen Arbeitsplatzrechner für den professionellen Einsatz, der in seinen technischen Parametern den Heim- und Personalcomputern weit überlegen war. Ein charakteristisches Merkmal der Workstation ist vor allem ein grafischer Bildschirm mit Einzelpunktansteuerung, auf dessen Basis ein fortgeschrittener Mensch-Maschine-Dialog geführt werden kann. Weitere Geräte der Grafikperipherie wie Flachbett- oder Rollenplotter, Rollkugeln, Digitalisiertabletts und Mäuse vervollständigen oft eine Workstation.
Auf Grund des weltweiten Übergangs zur breiten Anwendung von rechnergestützten Methoden und Verfahren wie Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Büroautomatisierung mit anspruchsvoller Geschäftsgrafik, Desk-Top-Publishing u. v. a. m. entwickelte sich der Workstation-Markt besonders in den letzten Jahren rasant. Alle führenden Rechnerhersteller bieten zu ihren klassischen Klein- und Großrechnern entsprechende Workstations an. Manche Hersteller haben sich sogar nur auf Workstations spezialisiert. Die Tabelle zeigt, wie sich die Marktanteile (in %) bei den verkauften Stückzahlen weltweit auf die führenden Firmen aufteilen (Quelle Dataquest, IDC).

Hersteller	1986	1987
Sun Microsystems	26	28
Apollo Domain	24	21
Hewlett-Packard	22	16
Digital Equipment	8	21
Andere	20	14

Workstations werden in der Regel als „intelligente“ Datenendplätze von (Hochleistungs-) Klein- oder Großrechnern verwendet, wodurch der Workstationbenutzer bei Bedarf auf vom Zentralrechner verwaltete, gemeinsam genutzte Datenbanken, eine größere Verarbeitungsleistung und spezielle Peripherie wie Laserdrucker, Magnetbandarchive usw. zugreifen kann.

Entwicklungstrends

Ausgangspunkt der Workstationentwicklung war die Entlastung der teuren Zentralrechner von den zeit- und speicherplatzaufwendigen Grafikoperationen, die die Leistung der Zentralrechner erheblich minderten und im Multiuserbetrieb die Qualität des Mensch-Maschine-Dialogs durch unvertretbare Wartezeiten beträchtlich minderten. Durch Verlagerung der elementaren Grafikoperationen (Pan, Zoom, Clipping, Rotation usw.) auf einen preiswerten Mikrorechner in das Grafikterminal wurde der Zentralrechner erheblich entlastet. Mit der weiteren Entwicklung der Workstation-Technologie vergrößerten sich ständig die Forderungen nach immer mehr Computerleistung, nach größeren Bildschirmen und immer höherer Auflösung der Bildschirme am Arbeitsplatz des Ingenieurs, Kartographen oder Redakteurs. Mit Realisierung eines effektiven Transfers der Daten und Grafiken zwischen Zentralrechner und Workstation hat sich auch ständig die Forderung nach schnelleren Kommunikationsmedien erhöht, so daß die Entwicklung der Workstation-Technologie in enger Einheit mit der Entwicklung der LAN-Technologie zu sehen ist.
Betrachtet man heute das Angebot an Workstations auf dem Weltmarkt, muß man feststellen, daß die Produktpalette sehr breit ist und die Grenzen bei dem rasanten Entwicklungstempo stark fließend sind. In das Gebiet der Workstations fließen verstärkt die leistungsstarken Personalcomputer wie die Personalcomputer des IBM Personal System/2 (Modell 50, 60, 70 und 80) als auch Workstations auf der Basis von klassischen Kleinrechnern wie VAXstation 8000 und auf der Basis von Supermikrorechnern wie VAXstation 3200 und 3500 ein.

Definition

Wird die Frage nach einer Definition des Begriffes „Workstation“ gestellt, dann erweist sich bei dem gegenwärtigen enormen Entwicklungstempo die Beantwortung als recht kompliziert. In den USA wurde vor etwa 5 Jahren der Begriff einer 5M-Maschine geprägt, worunter man die Mindestanforderungen verstand:

- 1 MIPS (Million Befehle/Sekunde)
- 1 MByte Arbeitsspeicher
- 1 Million Bildpunkte (Pixel)
- 1 MBit/s Datenübertragungsrate
- Mehrfenstertechnik

Diese Anforderungen waren vor einigen Jahren für viele Systeme noch unüberwindbar. Heute werden sie aber schon von Personalcomputern und teilweise von Heimcomputern erreicht. Seit 2 Jahren werden verstärkt auf den Markt Superworkstations auf der Grundlage der RISC-Architektur (Reduced Instruction Set Computer) gebracht, die eine Verarbeitungsleistung bis zu 10 MIPS erreichen.
Die RISC-Technologie mit dem eingeschränkten Befehlssatz ermöglicht durch den Wegfall der Mikroprogrammierung im Mikroprozessor die Produktion von sehr schnellen, preiswerten Mikroprozessoren. RISC-Workstations zeichnen sich damit durch ein wesentlich besseres Preis-/Leistungsverhältnis aus, was zu einem weiteren starken Preisverfall führte. So werden heute Workstations bereits für 10000 DM von verschiedenen Herstellern angeboten.
Von Hewlett-Packard wurde 1986 die erste RISC-Architektur, die sogenannte Precision Architecture, einheitlich für Workstations und Minicomputer angekündigt. Entsprechende Workstations HP 9000/825 SRX und HP 9000/835 SRX wurden vor einem Jahr vorgestellt. Sun Microsystems entwickelte einen eigenen RISC-Mikroprozessor, der in der neuen Workstation Sun 4/200 eingesetzt wird, die die bisher sehr erfolgreiche Workstation-Familie Sun 3 ablösen wird. Von Motorola wurde weiterhin speziell für Workstations der 32-Bit-RISC-Prozessor MC 88000 auf den Markt gebracht.
Natürlich kann eine Rechnerleistung von 1 MIPS eines RISC-Prozessors keinesfalls mit 1 MIPS einer MicroVAX verglichen werden, aber betrachtet man das Aufgabenprofil einer Workstation, dann überwiegen eindeutig die kurzen und einfachen Operationen, so daß die RISC-Architektur klar überlegen ist.

Workstation-Typen

Bei den Workstations der oberen Leistungsklasse spricht man heute oft schon von der 10-M-Maschine, de-

Software

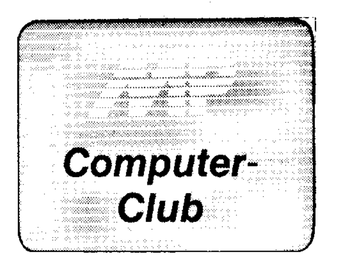
Mit den 32-Bit-Mikroprozessoren bzw. RISC-Prozessoren hat sich eindeutig als Basisbetriebssystem für Workstations durchgesetzt. Die meisten Hersteller wie Apollo orientierten dabei auf Systeme, die zum Berkeley-UNIX BSD 4.3 kompatibel sind. Als zweite Hauptrichtung sind verstärkt UNIX-Systeme entsprechend der System-V-Interface-Definition von AT&T bzw. den Empfehlungen der X/OPEN-Gruppe anzusehen.
Auf den leistungsschwächeren Workstations wird für technische Anwendungen vorrangig die 2D-GKS-Software und auf den leistungsstärkeren Workstations die 3D-GKS-Software angeboten. Für die Windowtechnik wird allgemein Software auf der Basis des X11-Window-Standards bereitgestellt.

Haupteinsatzgebiete

In der Vergangenheit war das Haupteinsatzgebiet von Workstations vor allem das rechnergestützte Konstruieren (CAD) auf fast allen ingenieurtechnischen Gebieten.
Wichtige Anwendungen sind vor allem der Schaltkreis- und Leiterkartenentwurf, wobei die Schaltkreis- und Logiksimulation meist auf einem leistungsfähigeren Zentralrechner ausgeführt wird. Frühzeitig hatte sich auch das Gebiet des Entwurfs von mechanischen Konstruktionen entwickelt, wobei auch hier die Festigkeitsberechnungen auf der Basis der Finite-Elemente-Methode (FEM) auf einen sehr leistungsfähigen Zentralrechner delegiert werden.
Zu den neueren Anwendungsgebieten gehören vor allem:
- rechnergestützte Softwareentwicklung (CASE – Computer aided software engineering)
- Expertensysteme (KI-Workstations) für fast alle Wissensgebiete
- Büroautomatisierung
- Desk-Top-Publishing

Zweifellos sind damit noch lange nicht alle Anwendungsgebiete erschlossen. Aber mit der weiteren Entwicklung der Workstation-Technologie werden durch diese leistungsfähigen Arbeitsplatz-Rechner zunehmend die Arbeitsinhalte nahezu aller Berufsgruppen tiefgreifenden sozialen Veränderungen unterzogen.

Prof. Dr. Thomas Horn



Kurzes Maschinenprogramm mit großem Effekt

Dr. Lothar Boltze
Martin-Luther-Universität Halle-Wittenberg, Sektion Mathematik

Die algorithmische Aufbereitung der Problemstellung bei der Entwicklung eines Programms ist die halbe Lösung!

Dieser Beitrag wendet sich an Kleincomputerprogrammierer mit Interesse für Maschinencodeprogramme.

Es soll ein Programm entwickelt werden, mit dessen Hilfe alle Kommentare und unnötigen Leerzeichen in einem BASIC-Programm der KC 85-Familie beseitigt werden können. Dazu ist es notwendig zu wissen, wie ein BASIC-Programm im Speicher des Rechners aufgebaut ist. Die Struktur ist in Tafel 1 angegeben. Sie wird auch als physische Programmstruktur bezeichnet. Sowohl logisch als auch physisch stellen die BASIC-Programme der KC 85-Familie lineare Listen dar. Verbal läßt sich eine Datenstruktur, die als lineare Liste bezeichnet wird, wie folgt beschreiben:

- Ein Datenelement (= BASIC-Zeile) ist das letzte oder besitzt einen wohldefinierten Nachfolger.
- In Verbindung mit BASIC-Programmen ist das nachfolgende Datenelement diejenige Zeile mit der kleinsten Zeilennummer, die größer als die Zeilennummer der betrachteten Zeile ist.

Die symbolischen Adressen BKPFZ, DSTi (i = 1, 2, 3) in Tafel 1 sind BASIC-Notizspeicheradressen und stehen entsprechend für BASIC-(Programm-)Kopfzeiger und Datenstapelzeiger.

Der Grundgedanke für die Umsetzung des Problems in einen Algorithmus besteht darin, das BASIC-Programm, beginnend am Programmanfang auf *den* Speicherbereich, unter Übergehen von Kommentaren und unnötigen Leerzeichen, zu kopieren, den es nach dem Laden im OK des BASIC einnimmt. Besonders günstig läßt sich der Grundgedanke in der Assemblersprache für den U 880 unter Verwendung des LDI-Befehls realisieren. Dem wurde schon bei der Erarbeitung des Algorithmus Rechnung getragen. Den Algorithmus geben wir in Form eines Programmablaufplanes an (siehe Bild 1). Das Assemblerprogramm (Bild 2) wurde anhand des Programmablaufplanes für die Kleincomputer KC 85/1 und KC 87 erarbeitet. Das entsprechende Maschinencodeprogramm wird im OS geladen und mit dem Namen 'NCOMMENT' im OS aufgerufen (siehe auch /1/). Der dem Algorithmus entsprechende Maschinencode ist verschiebbar und kann sofort auf KC 85/2 und KC 85/3 implementiert werden. Es sei darauf hingewiesen, daß im Algorithmus nicht berücksichtigt wurde, daß manche Programmierer Zeichenkettenkonstanten nicht ordnungsgemäß abschließen, wenn sie am Ende einer BASIC-Zeile stehen.

Das betrachtete Problem eignet sich auch für eine Behandlung in einem Informatik-Schülerzirkel. Dabei ist es gut geeignet, der allgemein beobachtbaren Tendenz, Programment-

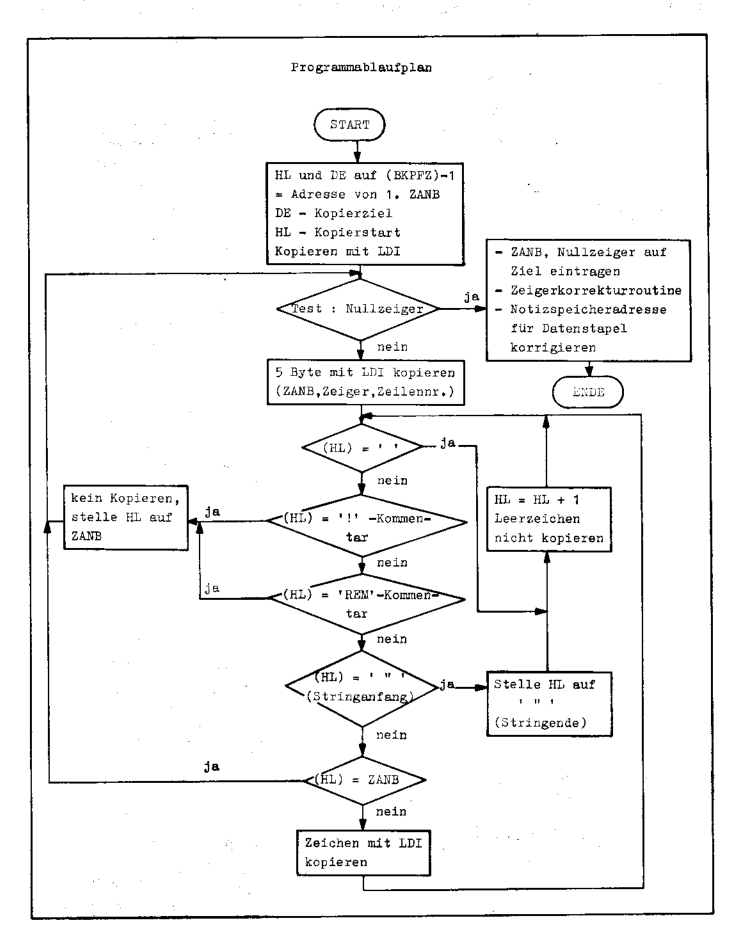


Bild 1 Programmablaufplan

Tafel 1 BASIC-Programmstruktur

RAM-Adresse	Inhalt	Bemerkungen
a - 1	0	Zeilen-Abschluß-Null-Byte (=ZANB)
a:=(BKPFZ,BKPFZ+1)	☐	Zeiger; gibt an, auf welcher RAM-Adresse die nächste BASIC-Zeile beginnt, (a,a+1) = (a+1)*256+(a) =: a1
a + 1	☐	
a + 2	☐	BASIC-Zeilennummer = (a+3)*256 + (a+2)
a + 3	☐	
	.	Quelltext der BASIC-Zeile
a1 - 1	0	ZANB
a1 := (a,a+1)	☐	Zeiger
	☐	
	☐	Zeilennummer
	☐	
	.	
	0	ZANB (letztes)
	0	Nullzeiger = BASIC - Programmende
	0	
(DST1,DST1+1) --->	☐	Datenstapel, wird bei Programmabarbeitung erzeugt
(DST2,DST2+1) --->	.	
(DST3,DST3+1) --->	.	

(a) bezeichnet den Inhalt der Speicherzelle a.

```
LOC   CODE     LINE       SOURCE

                                PN    NCOMMENT
                         ;
                         ;Programm zum Beseitigen aller Kommentare und
                         ;unnoetigen Leerzeichen in BASIC-Programmen der
                         ;KC 85 - Familie.
                         ;
                         ;Implementierung fuer KC 85/1 und KC 87.
                         ;
                                ORG   7E00H
                         ;
                         ;Definitionen
                         ;-------------
                         COM:   EQU   9CH              ;='!'-Kommentar-Token
                         REM:   EQU   8EH              ;=REM-   - '' -
                         BKPFZ: EQU   863              ;BASIC-Kopf-Zeiger
                         DST1:  EQU   983              ;Datenstapelanf.
                         DST2:  EQU   985              ; -''- indizierte Var.
                         DST3:  EQU   987              ;USR(.)-Notizanf.
                         ;
7E00 C30C7E                     JMP   NCOM             ;Sprungtabelle fuer
7E03                            DB    'NCOMMENT'       ;Aufruf im OS
7E0B 00                         DB    0
                         ;
7E0C C5                  NCOM:  PUSH  BC
7E0D D5                         PUSH  DE
7E0E E5                         PUSH  HL
7E0F F5                         PUSH  AF
                         ;
7E10 2A5F03                     LD    HL,(BKPFZ)       ;HL und DE auf ZANB
7E13 2B                         DEC   HL               ;vor Anf. der 1. Zeile
7E14 E5                         PUSH  HL               ;HL = Kopierstart
7E15 D1                         POP   DE               ;DE = Kopierziel
7E16 23                  WE:    INC   HL               ;HL auf 1. Zeigerbyte
7E17 7E                         LD    A,(HL)           ;TEST:Nullzeiger?
7E18 23                         INC   HL
7E19 B6                         OR    (HL)
7E1A 2837                       JRZ   ZDSTK-#          ;-> Restbehandlung
                         ;
7E1C 2B                         DEC   HL               ;HL auf zu ko-
7E1D 2B                         DEC   HL               ;pierendes ZANB
7E1E EDA0                       LDI                    ;ZANB kopieren
7E20 EDA0                       LDI                    ;Zeiger
7E22 EDA0                       LDI                    ;        kopieren
7E24 EDA0                       LDI                    ;ZNR
7E26 EDA0                       LDI                    ;        kopieren
                         ;Normales kopieren mit TEST auf ' ','!','REM','"'
                         ;bei Einstieg wird HL auf 1.Quelltextzeichen
                         ;der aktuellen BASIC-Zeile setzen.
                                DEC   HL
```

```
7E29 23       00050 NZ:    INC  HL              ;auf naechstes Zeichen
7E2A 7E       00051 ZEHO:  LD   A,(HL)          ;Zeichen holen
7E2B FE20     00052        CMP  ' '             ;Leerzeichen?
7E2D 28FA     00053        JRZ  NZ-#            ;uebergehen
7E2F FE9C     00054        CMP  COM             ;'!'-Kommentar?
7E31 2007     00055        JRNZ K3-#
              00056 ;
7E33 AF       00057 SZANB: XOR  A               ;Kommentar, suche ZANB
7E34 23       00058 SZA1:  INC  HL
7E35 BE       00059        CMP  (HL)
7E36 20FC     00060        JRNZ SZA1-#
7E38 18DC     00061        JR   WE-#            ;HL zeigt auf ZANB
              00062 ;
7E3A FE8E     00063 K3:    CMP  REM             ;REM-Kommentar?
7E3C 28F5     00064        JRZ  SZANB-#
7E3E FE22     00065        CMP  '"'             ;String-Anfang?
7E40 2808     00066        JRZ  SSTRE-#         ;-> suche String-Ende
7E42 FE00     00067        CMP  0               ;ZANB?
7E44 28D0     00068        JRZ  WE-#            ;weiter, naechste Zeile
              00069 ;
7E46 EDA0     00070        LDI                  ;Zeichen kopieren
7E48 18E0     00071        JR   ZEHO-#
              00072 ;
7E4A EDA0     00073 SSTRE: LDI                  ;String kopieren
7E4C BE       00074        CMP  (HL)            ;suche String-Ende
7E4D 20FB     00075        JRNZ SSTRE-#         ;noch nicht Sting-Ende
7E4F EDA0     00076        LDI                  ;Sting-Ende kopieren
7E51 18D7     00077        JR   ZEHO-#
              00078 ;
7E53 12       00079 ZDSTK: LD   (DE),A          ;letztes ZANB eintragen
7E54 13       00080        INC  DE
7E55 12       00081        LD   (DE),A          ;Nullzeiger
7E56 13       00082        INC  DE              ;eintragen
7E57 12       00083        LD   (DE),A
7E58 EB       00084        EX   DE,HL           ;Datenstapeladressen
7E59 23       00085        INC  HL              ;im BASIC-
7E5A 22D703   00086        LD   (DST1),HL       ;Notizspeicher
7E5D 22D903   00087        LD   (DST2),HL       ;aktualisieren
7E60 22DB03   00088        LD   (DST3),HL
              00089 ;
7E63 ED5B5F03 00090        LD   DE,(BKPFZ)      ;Zeigerkorrektur
7E67 62       00091 K2:    LD   H,D
7E68 6B       00092        LD   L,E
7E69 7E       00093        LD   A,(HL)          ;Test: Nullzeiger?
7E6A 23       00094        INC  HL
7E6B B6       00095        OR   (HL)
7E6C 280E     00096        JRZ  AUSST-#         ;Ausstieg
7E6E 23       00097        INC  HL
7E6F 23       00098        INC  HL
7E70 23       00099        INC  HL
7E71 AF       00100        XOR  A
7E72 BE       00101 K1:    CMP  (HL)            ;Suche ZANB
7E73 23       00102        INC  HL
7E74 20FC     00103        JRNZ K1-#
7E76 EB       00104        EX   DE,HL
7E77 73       00105        LD   (HL),E          ;Zeiger
7E78 23       00106        INC  HL
7E79 72       00107        LD   (HL),D          ;eintragen
7E7A 18EB     00108        JR   K2-#
              00109 ;
7E7C F1       00110 AUSST: POP  AF
7E7D E1       00111        POP  HL
7E7E D1       00112        POP  DE
7E7F C1       00113        POP  BC
7E80 C9       00114        RET
              00115
7E81          00116        END
```

Bild 2 Assemblerprogramm NCOMMENT

wicklung nur im Dialog mit dem Computer vorzunehmen, entgegen zu wirken.

Literatur

/1/ Betriebssystem Z 9001, KC 85/1, KC 87. VEB Robotron-Meßelektronik „Otto Schön", Dresden 1986

☒ KONTAKT ☏

Martin-Luther-Universität Halle-Wittenberg, Sektion Mathematik, Universitätsplatz 6, Halle, 4020; Tel. 622524

Rechnen mit komplexen Zahlen in BASIC

Es gibt nur wenige Programmiersprachen (z. B. Fortran IV), die als Variablentyp eine komplexe Zahl $K = k_1 + i \cdot k_2$ zulassen. Trotzdem ist natürlich die Rechnung mit komplexen Zahlen auch in anderen Programmiersprachen programmierbar. In diesem Beitrag werden Hinweise gegeben, wie das in BASIC möglich ist, wobei die Kenntnis der Rechenregeln mit komplexen Zahlen vorausgesetzt wird.

Um Speicherplatzbedarf und Rechenzeit zu minimieren und eine hohe Übersichtlichkeit des Programms zu gewährleisten, werden die komplexen Zahlen nur in ihrer kartesischen (nicht exponentiellen) Form verwendet. Dem Real- und Imaginärteil werden verschiedene Variable zugeordnet, wobei ihre inhaltliche Zusammengehörigkeit durch geeignete Maßnahmen zu gewährleisten ist. Empfehlenswert ist die Verwendung von Arrays und die lückenlose Belegung aufeinanderfolgender Feldvariablen mit dem Real- und zugehörigen Imaginärteil. Bei einem Basic-Interpreter, bei dem der niedrigste Index der Feldvariablen 0 ist, sind dann alle Realteile in einer geradzahlig indizierten Feldvariablen und die zugehörigen Imaginärteile in der um 1 höher indizierten, ungeraden Feldvariablen gespeichert. Es ist zu beachten, daß wenige große Arrays weniger Speicherplatz benötigen als zahlreiche kleine mit gleichviel Feldplätzen. Werden einfache Variablen verwendet, so empfiehlt sich eine Indizierung, z. B. ar für Real- und ai für Imaginärteil der komplexen Zahl **a**. Schließlich wird vereinbart, daß im Imaginärteil die imaginäre Einheit i (bei der komplexen Wechselstromrechnung üblicherweise j) stets als Faktor bzw. im Zähler steht. Nur so wird gewährleistet, daß das Vorzeichen des Imaginärteiles eindeutig interpretiert werden kann.

Damit sind alle Voraussetzungen zur Programmierung komplexer Rechnungen geschaffen. Um mit wenig Speicherplatz und Rechenzeit auszukommen, sollten die im Programm benötigten Rechenarten mit komplexen Zahlen mit der DEF FN-Anweisung am Programmanfang, zusammen mit den Feldern (DIM-Anweisung), definiert werden. Gemäß den Rechenregeln für komplexe Zahlen und für Computer, deren BASIC-Interpreter mehrere Variablen in selbstdefinierten Funktionen zulassen, lauten diese Funktionen (A, C – Realteile und B, D – Imaginärteile der zu verknüpfenden komplexen Zahlen):

```
10 DEF FN S(A,B,C,D) = A + C:
   REM Realteil der Summe
20 DEF FN T(A,B,C,D) = B + D:
   REM Imaginärteil der Summe
30 DEF FN D(A,B,C,D) = A − C:
   REM Realteil der Differenz
40 DEF FN E(A,B,C,D) = B − D:
   REM Imaginärteil der Differenz
```

Die in den Zeilen 10 bis 40 definierten Funktionen bringen außer einer Erhöhung der Übersichtlichkeit Nachteile an Speicherplatzbedarf und Rechenzeit gegenüber der direkten Programmierung der Addition bzw. Subtraktion an der Programmstelle, wo der Funktionsaufruf erfolgen würde. Anders bei den folgenden Zeilen:

```
50 DEF FN O(A,B,C,D)
   = A * C − B * D:
   REM Realteil des Produktes
60 DEF FN P(A,B,C,D)
   = B * C + A * D:
   REM Imaginärteil des Produktes
70 DEF FN Q(A,B,C,D)
   = (A * C + B *
   D)/(C * C + D * D):  REM Realteil des Quotienten
80 DEF FN R(A,B,C,D)
   = (B * C − A *
   D)/(C * C + D * D):
   REM Imaginärteil des Quotienten
90 DEF FN K(A,B)
   = A/(A * A + B * B):
   REM Realteil des Kehrwertes
100 DEF FN L(A,B)
   = −B/(A * A + B * B):
   REM Imaginärteil des Kehrwertes
110 DEF FN B(A,B)
   = SQR(A * A + B * B):
   REM Betrag der komplexen Zahl
120 DEF FN C(A,B)
   = ATN(B/A):
   REM Winkel zwischen Real- und
   Imaginärteil
```

A∗A benötigt wesentlich weniger Rechenzeit als A↑2. Der Kehrwert (Zeilen 90 und 100) ist natürlich ein Sonderfall der Division (Zeilen 70 und 80 mit A = 1 und B = 0). Mit diesem Instrumentarium sind alle Verknüpfungen zweier komplexer Zahlen möglich. Sind mehrere komplexe Zahlen miteinander zu verrechnen, so wird die Rechnung in mehrere Rechenschritte gegliedert und das Zwischenergebnis in zwei dafür vorzusehende Feldvariablen oder einfachen Variablen gespeichert. In manchen Fällen wird es auch möglich sein, die für das Endergebnis vorgesehene (Feld-) Variable vorerst für das Zwischenergebnis zu benutzen und anschließend mit dem Endergebnis zu überschreiben. In Programmen, in denen bestimmte Verknüpfungen mehrerer komplexer Zahlen häufig vorkommen, können dafür weitere Funktionen selbst definiert werden, wobei darin die angegebenen vorkommen dürfen. Ist beispielsweise die Zeile 50 im Programm enthalten, kann Zeile 70 auch lauten:

```
70 DEF FN Q(A,B,C,D)
   = FN O(A,B,C,D)/(C * C + D * D)
```

was hier aber keine Vorteile bringt. Die Rechnung wird durch Aufruf der FN-Anweisung ausgeführt, z. B. berechnet:

```
130 DIM A(5)
200 LET A(4) = FN O(A(0),A(1),A(2),
    A(3))
210 LET A(5) = FN P(A(0),A(1),A(2),
    A(3))
```

das Produkt der in A(0) bis A(3) gespeicherten komplexen Zahlen und legt den Realteil in A(4) und den Imaginärteil in A(5) ab.

Einige Heim- und Personalcomputer (z. B. KC 85, Commodore) lassen in selbstdefinierten Funktionen nur eine Variable zu. Dieser Nachteil wird folgendermaßen kompensiert: In den Zeilen 10 bis 120 erscheint in der Klammer hinter dem Funktionsnamen nur eine Variable, z. B. A für den Realteil der ersten Zahl, alles andere bleibt unverändert. Vor dem Funktionsaufruf müssen den restlichen in der Funktion vorkommenden Größen, die als Konstante interpretiert werden, mit der LET-Anweisung die aktuellen Werte zugewiesen werden. Somit ergibt sich:

```
200 LET B = A(1): LET C = A(2):
    LET D = A(3): LET A(4) = FN
    O(A(0))
210 LET A(5) = FN P(A(0))
```

Ein gutes, aber nicht ausreichendes Kontrollinstrument für die Programmtestung ist das Setzen aller Imaginärteile der Ausgangswerte auf Null. Dann müssen alle Ergebnisse und Zwischenergebnisse ebenfalls reell sein. Der umgekehrte Fall gilt nicht, wie schon an der Produktbildung (Zeile 50) zu sehen ist!

Wolf-Dieter Krohs

Ziffernprüfung

Häufig ist es erforderlich, numerische Variable nicht numerisch, sondern in Strings abzuspeichern (z. B.: Artikelnummern, Materialnummern, Schlüsselnummern usw.).

Die Option „PICTURE" läßt bei der Maskierung mit „#" oder „9" außer Ziffern auch Sonderzeichen wie „+", „–", „.", „ " zu.

Einen String daraufhin zu überprüfen, ob nur Ziffern vorhanden sind, ist normalerweise nur zeichenweise möglich.

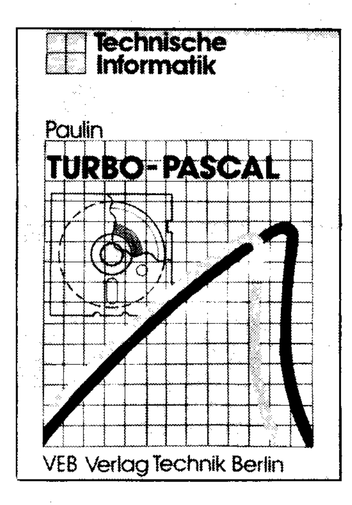
Die folgende Befehlssequenz ist schneller. Dabei ist die Länge des zu überprüfenden Strings auf maximal 9 Stellen begrenzt.

Thomas Steffens

```
*---------------------------------------------------------
* PRUEFNUM.PRG / 27.05.87 / PRUEFEN VON ZIFFERNSTRINGS
*---------------------------------------------------------
* EINGANG: ME1 = < ZU PRUEFENDER STRING >
*
* INTERN : MI1 = < ZWISCHENSPEICHER >
*
* AUSGANG: MA1 = < T,WENN ALLES ZIFFERN / F,WENN NICHT >
*---------------------------------------------------------
IF LEN(ME1)<10
 IF '1'+ME1=STR(VAL('1'+ME1),LEN('1'+ME1))
  STORE T TO MA1
 ELSE
  STORE F TO MA1
 ENDIF
ELSE
 @ 21,0 SAY'STRING ZU LANG'+CHR(22)+CHR(7)
ENDIF
RETURN
*---------------------------------------------------------
```

TURBO PASCAL

von Gerhard Paulin
Reihe Technische Informatik, VEB Verlag Technik Berlin, 1988, 206 Seiten, 20 Bilder, 3 Tafeln, Broschur, 20,– M



Das moderne Sprachkonzept von PASCAL in Verbindung mit einer leistungsfähigen und unkomplizierten Entwicklungsumgebung hat dem TURBO-PASCAL-System zu großer Popularität verholfen. Der VEB Verlag Technik trägt dem mit vorliegendem Band Rechnung.
Dem Autor scheint trotz des geringen Buchumfangs (206 Seiten) ein Querschnitt gelungen zu sein, der einen breiten Leserkreis anzusprechen vermag. Hier findet sowohl der Einsteiger als auch der Fortgeschrittene wertvolle Anregungen. Der Schwerpunkt liegt neben einer im ersten Teil konzentrierten didaktischen Darlegung grundsätzlicher Aspekte von TURBO PASCAL vor allem auf der exemplarischen Demonstration der Möglichkeiten der Sprache PASCAL. So finden sich neben didaktisch orientierten Beispielen und Aufgaben eine Reihe für die Praxis nicht uninteressanter programmtechnischer Realisierungen – wie etwa: die numerische Integration mittels Simpsonscher Regel, die Lösung eines linearen Gleichungssystems oder die Listenverarbeitung, um nur einige zu nennen.
Auf den ersten Seiten werden Hinweise zur Handhabung der TURBO-PASCAL-Entwicklungsumgebung gegeben. Danach folgt eine systematische und auch für Anfänger geeignete Darstellung der Programmiersprache. Die praktische Anwendung der vorgestellten Sprachelemente wird stets an nachvollziehbaren Beispielen gezeigt. Behandelt und mit Programmquelltext illustriert werden weiterhin das Filekonzept, die Nutzung dynamischer und Pointer-Variablen, Recordvarianten sowie rekursive Prozeduren. Bei den Recordvarianten vermißt man lediglich die sogenannten „free unions". Dafür werden die „discriminated unions" ausführlich und allgemeinverständlich behandelt. Eine hohe Anschaulichkeit bietet auch die Darstellung des Zeigerkonzeptes. Hier werden alle wesentlichen Methoden, die Nutzung des Pointer-Typs betreffend, mit Hilfe von Grafiken und Programmkonstrukten vorgestellt.
Zwölf Seiten sind der maschinennahen Programmierung mittels der TURBO-PASCAL-Implementation für CP/M-80 gewidmet. Es wird Grundsätzliches über externe Unterprogramme sowie INLINE-Assembler mitgeteilt. Auch der mit der Assemblerprogrammierung weniger Vertraute erhält einen Überblick über Mittel und Methoden hardwarenaher Programmierung in der TURBO-PASCAL-Implementation für CP/M-80. Alle Programmbeispiele und Ausführungen im Buch beziehen sich im übrigen auf diese Implementation. In einem der letzten Gliederungspunkte wird ein kurzer Überblick über Bildschirmprogrammierung und Bildschirmsteuerung sowie Grafikerweiterung der TURBO-PASCAL-Implementation für MS-DOS gegeben. Daran schließen sich die Lösungen inclusive Besprechung der in vorangegangenen Gliederungspunkten gestellten Aufgaben an. Damit ist die Möglichkeit der Vertiefung von Kenntnissen in der Programmiersprache im Selbststudium gegeben.
Ein Anhang (28 Seiten) enthält Zusammenstellungen der Syntax von TURBO PASCAL, der Standardfunktionen und -prozeduren sowie der Editorkommandos der integrierten Entwicklungsumgebung. Weiterhin finden sich eine ASCII-Code-Tabelle, eine Zusammenstellung der Fehlernachrichten des Compilers und der Laufzeitfehlermeldungen. Im letzten Teil des Anhangs ist ein Programmbeispiel für einen PASCAL-Quelltext-Lister abgedruckt, der alle reservierten Worte markiert. – Alles in allem ein gelungener Abschluß, der zum Weitermachen anregt. Dem Autor ist es mit „TURBO PASCAL" gelungen, ein didaktisch orientiertes, das TURBO-PASCAL-Handbuch ergänzendes, Arbeitsmittel zu schaffen, das auch dem täglichen Nutzer – ob Einsteiger oder Fortgeschrittener – eine Hilfe sein kann. *Th. B.*

Neue Technik – alte Gesellschaft

Silicon Valley
von W. Rügemer, Verlag die Wirtschaft Berlin 1986 (Lizenzausgabe des Pahl-Ruggenstein-Verlages Köln), 252 S., 27 Tab., 12 Bilder, 12,40 M

Der volle Untertitel dieser bemerkenswerten Broschüre lautet: „Silicon Valley: Mythos und Realität vom American Way of Technology". Es berichtet ein „Insider", ein engagierter Gewerkschaftler der BRD. Er schreibt fast wie Jack London – über seinen Studienaufenthalt im kalifornischen „High-Technology"-Mekka Silicon Valley, dem „Silizium-Tal" der USA. Es ist auch das Musterbeispiel der Organisation einer Hochtechnologie in der bürgerlichen Gesellschaft. Mit einer „neuen Leistungselite" ist es zugleich ein Beispiel der Talenteförderung und -ausblutung mit allen Mitteln kapitalistischer Wolfsgesetze, ein Beispiel der neuen Vermarktung wissenschaftlicher Erkenntnisse, also des Wissenstransfers von den Hochschulen und Universitäten der USA in die hochprofitable Privatwirtschaft. Das Schlüsselwort ist „hire and fire" – also „Anheuern", wenn es sich lohnt und „Feuern", wenn nicht mehr genügend dabei herausspringt. Werner Rügemer geht diesem Leben auf den *sozialen* Grund, untersucht erlebnisreich und mit vielen Detailbetrachtungen die Infrastruktur dieser Gesellschaft, der Armen, der Reichen und der Minderheiten, die Ausbeutung der Arbeitskraft, die Rolle der Schulen und Hochschulen, die Gefährdung der Umwelt, Kapitalinteressen und Kapitalstrategien, die Investitionen und Profite, den Niedriglohn, die Arbeitslosigkeit – aber auch die Solidarität weltanschaulich aufgeklärter und politisch denkender Menschen in diesem berühmt-berüchtigten Gebiet von etwa 1000 km^2 mit fast 2000 Firmen bei einer Dreiviertelmillion Menschen. Werner Rügemer ist kein Mikroelektroniker – das spürt man sofort, aber er ist ein bewundernswerter Analytiker des sozialen Geschehens, das den Spezialisten der Elektronik sonst verborgen bliebe. Dieses „Silicon Valley" leistet technologisch Beispielloses, das ist dem Fachmann gegenwärtig.
Um so bemerkenswerter ist die Leistung, die mit der Broschüre vorgelegt wurde. Zugleich ist es ein wichtiges geschichtliches Dokument zur amerkanischen Wirklichkeit.

Prof. Dr. Dr. M. Roth

Das große Computer-Viren Buch

von Ralf Burger, Data Becker GmbH, Düsseldorf, 1987, 363 Seiten, ISBN 3-89011-200-5

Dieses Buch von Ralf Burger bietet einen Einblick in einen Bereich, der für die meisten noch ein terra incognita darstellt. Mit dieser Einführung versehen, ist der interessierte Leser in der Lage, sich durchaus ein Bild von der Virenproblematik zu verschaffen.
Mit einem Überblick über die bisher bekannten und die noch denkbaren Virentypen und ihre Funktionsweisen ermöglicht der Autor, übrigens vom Standpunkt des gestandenen Programmierers ausgehend, eine Einschätzung der Bedrohung der Rechnerwelt durch die Computerviren. Unter Zuhilfenahme einer Studie von Stephan Ackermann legt der Autor auch eine gründliche Analyse der (immer noch sehr verworrenen) Rechtslage auf dem Gebiet der Computer- bzw. Datensicherheit in der BRD, aber auch international vor. Seine Standpunktfindung, motiviert von eigenen rechtlichen, ethischen und kommerziellen Gesichtspunkten, erörtert er überzeugend. Mit diesem Stil der Publikation bewegt sich der Autor sehr geschickt auf dem sehr schmalen Grat der Darlegung von Software und Soft War. Als bemerkenswert ist zu beurteilen, daß der Ansatz des Vergleiches von Computerviren mit Viren der biotischen Welt anschaulich gelungen ist.
Die umfassenden Beispiele illustrieren deutlich die Gefahr von „verseuchten" Systemen und ihre Wechselbeziehung. Demonstrationsprogramme lassen einige Wirkmechanismen erkennbar werden. Der Autor zeigt Infektionswege und damit gewisse Schutzmöglichkeiten bzw. -strategien und gelangt zum Fazit: Es ist kein wirkungsvoller umfassender Schutz gegen Computerviren außer der persönlichen Moral des Programmierers möglich. Die Gestaltung des Buches ist gut, die Themen werden übersichtlich angeordnet. Alles in allem läßt sich sagen: Das Buch bringt einen guten Überblick zum aktuellen Stand der Problematik, aber für einen Profi keine neuen bzw. weiterführenden Erkenntnisse.

Bernd Mischke

Problemlösen mit PROLOG

von M. Hanus, Verlag B. G. Teubner, Stuttgart, 1987, 224 S.

Die logische Programmierung gewinnt im Bereich der Künstlichen Intelligenz und hier insbesondere beim Entwurf von Expertensystemen, aber auch als allgemeines Programmierprinzip, zunehmende Bedeutung.
Das Buch aus der Reihe MikroComputer-Praxis gibt eine gut verständliche Einführung in das Konzept der Programmiersprache PROLOG und in ihre theoretischen Grundlagen, vermittelt eine umfassende Übersicht zu den Sprachbestandteilen und erläutert Programmiertechniken und Anwendungen anhand von Beispielen. Es ist gut geeignet für alle Leser, die PROLOG kennenlernen wollen. Dabei ist die Kenntnis anderer Programmiersprachen und der inneren Funktion von Computern nicht Voraussetzung.
Nach einer einleitenden Betrachtung zum grundsätzlichen Vorgehen beim Programmieren in PROLOG folgt eine systematische Einführung in die Programmiersprache.
Hierbei werden auch das Prinzip des Beweisens von Aussagen durch Resolution und die Unifikation als zentrale Mechanismen der logischen Programmierung herausgearbeitet. Elementare Programmiertechniken werden an Beispielen erläutert, aus denen der Leser auch einen Eindruck über den Anwendungsbereich von PROLOG gewinnt.
Anschließend wird auf die mathematischen Grundlagen der Programmiersprache eingegangen, wobei es gut gelungen ist, Anschaulichkeit und Exaktheit miteinander zu verbinden. Ein Vergleich der rein logischen Semantik mit der Semantik von PROLOG-Systemen wird vorgenommen.
In einer Übersicht werden weitere Sprachkonstrukte wie z. B. für die Ein-/Ausgabe, Arithmetik, Steuerung der Abarbeitung sowie vordefinierte Prädikate und Operatoren behandelt. Abschließend werden in zwei Abschnitten umfangreichere Anwendungsbeispiele vorgestellt, wobei die Anwendung von PROLOG auf die Beschreibung von Grammatiken hervorzuheben ist.
Eine wertvolle Hilfe für den Leser sind die zu jedem Abschnitt angegebenen Aufgaben und die Zusammenstellung der Lösungen sowie die Hinweise für die praktische Programmierung.

Dr. Harald Killenberg

Koppelprogramm PRG 600–PC 600 zur Prozeßüberwachung

Für das Programmiergerät PRG 600 des VEB Numerik „Karl Marx" wurde ein Koppelprogramm entwickelt, das eine Kommunikation zwischen PRG 600 (PRG 700) und der Steuerung PC 600 (PRG 700) während der Abarbeitung des eigentlichen Steuerprogramms gewährleistet. Die Prozeßüberwachung schließt u. a. folgende Arbeitsmöglichkeiten menügeführt ein:

• Anzeige von durch AD-Wandler erfaßten analogen Meßwerten auf dem Bildschirm
• Anzeige von Prozeßfehlern auf dem Bildschirm
• Änderung von Prozeß-Sollwerten im Steuerungsprogramm über Tastatur
• Archivierung von Prozeßzuständen und Meßwerten auf Diskette (ab Variante PRG 600/3). Zur Weiterverarbeitung der archivierten Werte (im vorliegenden Falle durch BASIC-Programme) am Bürocomputer A 5120/A 5130 können mit Hilfe eines Konvertierungsprogramms, das ebenfalls auf dem PRG implementiert ist, PRG 600-formatierte Disketten in SIOS-formatierte Disketten konvertiert werden.

Das komplexe Koppelprogramm wurde zur Prozeßüberwachung und Steuerung in verfahrenstechnischen Versuchsanlagen getestet und bewährt sich seit mehreren Jahren. Die im PRG 600 implementierten Programmodule besitzen weitgehend problemneutrale Schnittstellen und sind damit an unterschiedliche Problemstellungen anpaßbar.

VEB Komplette Chemieanlagen Dresden, Abt. 697030, PSF 184, Dresden, 8012; Tel.: 2321447

Dr. Syhre

Multitaskerweiterung rMTS für CP/M 86

Das System rMTS (Multi Task System) dient der Multitaskprogrammentwicklung im Betriebssystem CP/M 86. Die Vorteile gegenüber der Anwendung eines vollständigen Echtzeitbetriebssystems (z. B. BOS 1810) liegen in der Nutzung der weitverbreiteten CP/M-Programmentwicklungstechnik und der CP/M-Dateiorganisation sowie in dem geringen Speicherbedarf von rMTS (64 KByte).

Multitaskprogramme können in Assemblersprache oder in der Programmiersprache C entwickelt werden. Interfacebibliotheken gestatten die Anbindung von 56 NUCLEUS-Systemfunktionen (kompatibel zu RMX86). Unter CP/M 86 entwickelte Programmdateien (–.CMD– oder –.H86-Dateien) werden mit dem rMTS-Applicationloader geladen und als Tasks quasiparallel abgearbeitet. In den Tasks können CP/M-BDOS-Funktionen mit Einschränkungen genutzt werden. Die Konsolzugriffe in einem Multitaskprogramm werden über den rMTS-Konsolhandler organisiert.

Während der Programmentwicklung dient ein Objektdebugger der Diagnose der rMTS-Objekte.

rMTS läuft auf 8086-Rechnern unter CP/M 86 und SCP 1700 mit Zeitgeber- und Interruptlogik (8253, 8259A) und ≧ 128 KByte RAM (z. B. Single-Board-Computer 8086 des ZWGB der TU Dresden oder AC 7100). Interessenten erhalten eine 40seitige Dokumentation bei Zusendung einer 5¼"-2s2d-Diskette gegen eine Schutzgebühr von 50,– M.

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, Bereich Computertechnik, Albert-Einsteinstr. 2, Rostock, 2500

Dr. Jorke

V.24-IFSS-Umsetzer

Aufgabe war es, einen Hardware-Printserver zu schaffen, der 10 PC 1715 auf einen robusten Drucker 1152 ausgeben läßt. Dabei war das generelle Schnittstellenproblem V.24 ↔ IFSS (PC ↔ 1152) technisch sauber zu klären. Das heißt:

– galvanische Trennung aller Geräte
– zuverlässiger Aufbau (Schutzgrad)
– große Übertragungssicherheit
– einfache Handhabbarkeit.

Aus ökonomischer Sicht und Verfügbarkeitsgründen wurde eine Lösung mittels:

– Kanalauswahl über Drehschalter
– billigsten bipolaren Bauelementen
– separater Stromversorgung
– kompaktem Gehäuse (geschlossen)

realisiert, welche nachgenutzt werden kann. Es liegt ein kompletter Zeichnungssatz vor. Das Gerät ist nachbausicher und ohne Spezialwerkzeuge herstellbar. Eine klare Dokumentation erleichtert Aufbau und Inbetriebnahme.

IH Mittweida, Rechenzentrum, Platz der DSF, Mittweida, 9250

Kramer

Die Sprachwerkzeuge YACC und LEX unter DCP und MS-DOS

Bei der Verarbeitung formaler Sprachen (Programmier-, Fachsprachen, ...) sind die UNIX-Werkzeuge LEX und YACC effektive Hilfsmittel für den Programmierer, die aus einer Nutzerspezifikation ein C-Quell-Programm generieren: einen lexikalischen Analysator bzw. einen Parser. Der lexikalische Analysator durchmustert einen Dateistrom nach bestimmten Zeichenketten (Wörter), die vom Nutzer in der Form regulärer Ausdrücke spezifiert werden müssen. Die dabei erkannten Wörter werden dem Parser (als Morpheme) übergeben, der die syntaktische Analyse durchführt. Die der syntaktischen Analyse zugrunde liegende Grammatik wird vom Anwender in einer der Backus-Naur-Form ähnlichen Notation angegeben.

YACC und LEX haben über den Compilerbau im engeren Sinne Bedeutung für die Implementierung von Fach- und Beschreibungssprachen, z. B. in Experten- und Spezifikationssystemen. Die Sprachwerkzeuge YACC und LEX gehören zum Standardumfang eines UNIX-Betriebssystems. In der DDR standen sie bisher nur für Rechner mit UNIX-kompatiblen Betriebssystemen, z. B. für die Rechner K 1630 und A 71XX (Betriebssystem MUTOS) oder für P 8000 (Betriebssystem WEGA), zur Verfügung. Da DCP mehr Freiheiten zur Gestaltung der Bedienoberfläche bietet (direkter Zugriff auf Bildwiederholspeicher, Grafik, farbige Darstellung), ergeben sich hier Möglichkeiten, die Produkte der Sprachwerkzeuge LEX und YACC effektiv einzusetzen und einem großen Nutzerkreis zugänglich zu machen. Deshalb wurde eine Implementierung von LEX und YACC unter dem Betriebssystem DCP vorgenommen. Der Portierung der beiden Sprachwerkzeuge ging die Schaffung einer Möglichkeit zum effektiven Datentransfer zwischen UNIX und DCP voraus. Es wurde ein Funktionsäquivalent des UNIX-Dienstprogramms *tar* unter DCP entwickelt, womit der Datenaustausch zwischen DCP und MS-DOS sowie anderen UNIX-kompatiblen Betriebssystemen (z. B. MUTOS, WEGA u. a.) problemlos möglich ist. Es werden fast alle vom P 8000 verwendeten Diskettenformate (8 oder 9 Sektoren pro Spur, 40 oder 80 Spuren, ein- oder zweiseitig) unterstützt.

TH Ilmenau, Sektion INTET/TBK, Am Ehrenberg, PSF 327, Ilmenau, 6300; Tel. 74245, 74326

Dr. Rempel/Rudorfer

ERIKCOPY

ERIKCOPY ist ein Treiber für den Anschluß einer elektronischen Schreibmaschine vom Typ Erika S 3004 an den Kleincomputer KC 85/1 bzw. KC 87. Dabei wird weder der Druckermodul des KC noch die Interfacebox der Schreibmaschine benötigt. Die Umsetzung des ASCII-Codes in den Schreibmaschinen-Code erfolgt ausschließlich softwaremäßig im vorliegenden Programm. Das Programm wird in üblicher Weise geladen und durch 'ASGN LIST:=ERIKA' aufgerufen. Der Paralleldruck erfolgt nach gleichzeitigem Drücken der Tasten 'CONTR' und 'P'. Eine Hardcopy des Bildschirmes läßt sich durch zusätzliches gleichzeitiges Drücken der Tasten 'CONTR' und 'N' erzielen. Erfolgt der Aufruf durch

'ASGN LIST:=ERIKU'

so werden die Doppelbuchstaben A e usw. als Umlaute ausgegeben. Soll dies verhindert werden, so ist als Trennzeichen ein '&' dazwischen zu setzen. Weiterhin läßt sich das Zeichen 'ß' ausgeben. Dafür muß ein Klammerzeichen geschrieben werden.

Der Hardwareaufwand besteht im Verbinden des seitlichen PIO-Ports des KC mit der entsprechenden Buchse der Schreibmaschine über ein lediglich 2 Transistorstufen enthaltendes Anpaßglied.

VEB Magdeburger Armaturenkombinat, Abt. TKA, Postfach 330, Magdeburg, 3010; Tel. 4390/App. 497

Dr. Nestler

Modula-2-Entwicklungssysteme und portable Bibliothek

Zur Nachnutzung angeboten werden:

• Portable Modula-2-Systembibliothek
• Komplette Modula-2-Entwicklungssysteme mit Zielkodegenerierung für PDP-11 unter RSX 11M und Z 80-Computer unter CP/M oder UDOS. Die Entwicklungssysteme sind auch für kompatible Rechner und Betriebsysteme (OS-RW, MOOS1600, SCP, CP/A u. a.) nutzbar.
• Zusatzprogrammpakete.

Die Bibliothek sowie jedes der 3 Entwicklungssysteme stehen lauffähig und mit identischer Benutzeroberfläche unter jedem der genannten Betriebssysteme sowie zusätzlich unter MS-DOS und DCP als Original- bzw. Cross-Software zur Verfügung. Programme, die unter einem der oben genannten Betriebssysteme unter Nutzung der portablen Bibliothek entwickelt werden, sind auch unter allen anderen Betriebssystemen durch Neuübersetzung sofort verfügbar (100 % Quelltextportabilität).

Bibliothek

Zur portablen Bibliothek gehören z. Zt. folgende Komponenten:

– Alphanumerische, akustische und windoworientierte Terminal-Ein-/Ausgabe (Module: KeyIn, DispOut, TermIO, Terminal, WindowOut, Sound)
– Pixelorientierte Vollgrafikroutinen für unterschiedliche Grafikhardware (Geometry, Graphics)
– Datei-Ein-/Ausgabe für unstrukturierte, binäre und textstrukturierte Files sowie Verwaltungsoperationen für Dateiverzeichnisse (Files, Streams, Texts, InOut, FileOpen, DirEntries)
– Konvertierungsroutinen für Umwandlungen zwischen interner (binärer) und externer (formatierter) Datendarstellung für alle Datentypen (WriteNums, WriteReals, ReadNums, ReadReals, NumToStr, RealToStr, Radix50)
– Zeichen- und Stringverarbeitung (ASCIIDefs, Chars, Chars1, Strings)
– Zerlegung und Auswertung von Kommandozeilen (FileNames, Options, NameAndOptions)
– Einheitliche Auslösung, Behandlung und Anzeige von Laufzeit-, Bibliotheks- und Nutzerfehlern (Exceptions, ErrorMessages)
– Überlagerungslader zur Realisierung von Programmüberlagerungen im Hauptspeicher (Loader)
– Mathematische Standardfunktionen und Matrixgrundoperationen (MathLibO, MathConstO, Matrix)
– Ermittlung und Ausgabeformatierung von Zeit und Datum (Clock, Times)
– Testhilfekomponenten zur Unterstützung des Aufbaus dialogorientierter Testrahmen für Module sowie zur einfachen Realisierung von On-line-Testausgaben und zur Kodereassemblierung (T, W, DisAss80, DisAss86)
– Systemkonstanten (Const1)

Zusätzlich sind für die jeweiligen Betriebssysteme nichtportable, systemnahe Module vorhanden, die spezielle Leistungen des jeweiligen Betriebssystems bzw. Prozessors verfügbar machen.

Entwicklungssysteme

Das Entwicklungssystem für PDP-11, SM4, K1630 beinhaltet z.Zt. folgende Teile:
- Compiler RCO
- Linker RLI
- Lader RLO
- Symbolischer Hochsprach-Post-Mortem-Debugger RPD
- Taskgenerierprogramm LNK2TSK zum Linken von RSX11M-Tasks
- Klartextfehlermeldungsprogramm ERR
- Filedekoder RFD zur Dekodierung/Reassemblierung aller im System verwendeten Filetypen.

Die Entwicklungssysteme für CP/M und UDOS bestehen jeweils aus folgenden Komponenten:
- Compiler MCO
- Linker MLI
- Lader MLO
- COM-File-Generierprogramm LOD2COM
- PRG-File-Generierprogramm COM2PRG (nur UDOS)
- Klartextfehlermeldungsprogramm ERR
- Filedekoder MFD zur Dekodierung/Reassemblierung aller im System verwendeten Filetypen.

Zusatzprogrammpaket

- Modula-2-Crossreferenzlister XRF.

Die Programmiersysteme unter CP/M und UDOS arbeiten mit einem Zwischenkode, der interpretativ abgearbeitet wird. Dadurch erhöht sich die Kompaktheit des Kodes gegenüber Assemblerprogrammierung beträchtlich, aber die Abarbeitungsgeschwindigkeit verringert sich.

Für die Nutzung eines Entwicklungssystems unter CP/M bzw. UDOS ist das Vorhandensein einer RAM-Disk empfehlenswert, aber nicht Bedingung.

TU Dresden, Sektion 9 Informationstechnik, Bereich 6 Automatisierungstechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4635068 *Hauptmann*

Erstellung des betrieblichen Kennziffernspiegels

Zur Erstellung des betrieblichen Kennziffernspiegels und des Haushaltsbuchs wurde ein Programm unter dem Betriebssystem SCP mittels Datenbanksystem dBase II entwikkelt. Es wird ein Plan-Ist-Vergleich der Kennziffern aller Kostenstellen zum Jahresplan und zum Plan-Berichtszeitraum sowie die Entwicklung der Kennziffern im Vergleich zum gleichen Berichtszeitraum des Vorjahres durchgeführt. Das Programmsystem zeichnet sich durch seine offene Struktur aus. Eine Anpassung an die entsprechende Betriebsstruktur ist mit minimalem Aufwand realisierbar.

Die Anzahl der Kostenstellen, der Kennziffern sowie die Anzahl der Verknüpfungen von Eingabekennziffern ist beliebig und nur durch die Diskettenkapazität begrenzt.

Die Anzahl der Kostenstellen für jede einzelne Kennziffer ist variabel einstellbar, so daß nutzerseitig nur die tatsächlich benötigten Kostenstellen bearbeitet werden müssen.

Die Drucksteuerung ist so organisiert, daß bis zu 10 inhaltlich verschiedene Druckvarianten gleichzeitig angeboten werden.

VEB Landtechnische Instandsetzung Laucha, Golzenerstraße 6, Laucha/U., 4807 *Kirsten*

Parallelverarbeitung unter CP/M und Turbo-Pascal

Es wird eine Softwarelösung zur Nachnutzung angeboten, die eine Parallelverarbeitung unter CP/M-Betriebssystemen und Turbo-Pascal ermöglicht. Das Prinzip eignet sich für Applikationen, wo neben einem Hauptprogramm eine feste Anzahl von Hintergrundtasks bearbeitet werden müssen, z.B. bei Echtzeitanwendungen, Rechnerkopplung, Protokolldruck und Grafiksystemen. Kriterien sind:
- Installieren des Regimes *Parallelverarbeitung* für das CP/M automatisch beim Kaltstart des Systems
- Anlagerung der Steuerung für die Parallelbearbeitung im Betriebssystem ohne Verwendung von Interrupts
- Die Tasksteuerung läuft unabhängig vom Hauptprogramm im Hintergrund ab. Damit können beliebige CP/M-Anwenderprogramme ohne Modifizierung in die Parallelarbeit einbezogen werden. Die Steuerung der Parallelarbeit benötigt praktisch keine Rechenzeit, da Systemtotzeiten ausgenutzt werden.
- Haupt- und Hintergrundprogramme können in Turbo-Pascal oder Assembler geschrieben sein.

Zur Nachnutzung stehen zwei Softwarepakete zur Verfügung:

1. Parallelverarbeitung für CP/M (allgemeine Lösung)

Entsprechend Schnittstellenbeschreibung kann der Nachnutzer beliebige Parallelverarbeitungssysteme realisieren.

2. Spezielle Parallelverarbeitung für CP/A der AdW zur Kopplung mit Rechnern EAW electronic S2000-R über IFSS/V.24

Dieses Softwarepaket ermöglicht den Aufbau von Prozeßleitsystemen auf PC/BC unter CP/A und Turbo-Pascal. Dabei wird das Hauptprogramm in mehreren Hintergrundebenen im Sinne der Echtzeitverarbeitung von Kopplungsaufgaben zu maximal 8 Prozeßreglern S2000-R über IFSS/V.24 und Protokolldruck entlastet.

VEB Zentralinstitut für bezirksgeleitete Industrie und Lebensmittelindustrie, Abt. 114, Eichbuschallee 51, Berlin, 1195; Tel. 6329911 App. 65
Kalok/Klein/Kretzschmann

Universelle K-1520-Schnittstellensteuerung für Drucker und Plotter

Die Mehrzahl derzeit vorhandener Drucker verschiedener Hersteller verfügt über eine V.24- oder IFSS-Schnittstelle. Ebenfalls verbreitet sind parallele Schnittstellen wie Centronics und IFSP. Rückläufig ist der Einsatz der Seriendrucker SD 1154 und SD 1156. Die Steckeinheit DAP ermöglicht die Ansteuerung folgender Schnittstellen:
Centronics, IFSP, SIF 1000/1 (SD 1156), Sonderschnittstelle für SD 1154, V.24, IFSS.

Die Bedienung der seriellen Schnittstellen erfolgt mittels Adapter in Verbindung mit der STE DAP als Funktionseinheit. Die Schnittstellensteuerung übernehmen 2 PIO UB 855 unter Verwendung entsprechender Leitungstreiber. Zum Anschluß der parallelen Interfaces sind 39polige Stekkerleisten der Bauform 302-39 verwendet worden. Zur Parallel-Serien-Wandlung der Daten werden 2 Leitungen der PIO-Ports (Ein-/Ausgabe) verwendet. Die Datenübertragungsrate wird softwareseitig von der Programmlaufzeit abgeleitet. Damit steht dem Anwender eine sehr preisgünstige und leistungsstarke Schnittstellensteuerung unter minimalem Hardwareeinsatz zur Verfügung.

Die K-1520-Leiterplatte sowie die genannten Adapter können kurzfristig bezogen werden.

Im Softwareangebot stehen zur Nachnutzung bereit:
- Programme für die Gerätegeneration MC 80.xx zum Druck von Quelltext, Objektcode, BASIC-Listing, BASIC-Argumenten, Grafik usw.
- Driver (Initialisierung, Programm zum Druck eines Zeichens) für alle genannten Schnittstellen, angepaßt an den jeweiligen Druckertyp.

VEB Elektronik Gera, Abt. EN, Parkstraße 3, Gera, 6500; Tel. 584167
Ehrhardt

TMAS/GETVAR für Turbo-Pascal

Das Programm **TMAS** stellt ein Programmierwerkzeug für den Pascal-Programmierer dar. Es kann in der Hand sowohl von Anfängern als auch von fortgeschrittenen Programmierern bei der Erstellung bildschirmorientierter Ein-/Ausgabeprozeduren, Nutzermenüs oder nur einfacher Bildschirmmasken ohne Variablenausgabe einen hohen Rationalisierungseffekt erzeugen. Es handelt sich um einen bildschirmorientierten Programmgenerator, das heißt, fast alle Informationen, die der Generator benötigt, werden der Bildschirmdarstellung entnommen. Die Bedienung ist der von Turbo-Pascal angenähert (ebenfalls eigener Editor).

Von TMAS generierte Prozeduren sind vom Programmierer geeignet zusammenzustellen, um Variablendeklarationen und -initialisierungen zu ergänzen, damit ein compilierfähiges Programm entsteht. Nur selten wird geringfügiges Verändern notwendig sein.

Die Programmdatei **GATVAR. INC** enthält den Quelltext einer universellen, eventuell vom Nutzer auch erweiterbaren Eingabeprozedur, die auch unabhängig vom Generator in eigenen Programmen nutzbar ist. Eine ausführliche Bedienungsanleitung sowie 3 Beispiele gehören ebenfalls zum Nachnutzungsumfang.

VEB Robotron Büromaschinenwerk „Ernst Thälmann" Sömmerda, BfN, Weißenseer Str. 52, PSF 43, Sömmerda, 5230; Tel. 43257 (BfN), 21110 (Schenk) *Dr. Koch/Schenk*

Turbo-Window-Box für SCP 1700

Es wurde ein Programm entwickelt, das es ermöglicht, die von MS-DOS bekannte Windowtechnik auch unter SCP 1700 zu ermöglichen. Damit wird der Programmierer in die Lage versetzt, schnell und unkompliziert Menüs und Hilfsmenüs aufzubauen und den Bediener zu führen.

Die Window-Box ist speziell auf die Hardware (Bildschirm Alpha 1) des A 7100 zugeschnitten. Durch Einbinden der Window-Box in ein beliebiges Turbo-Pascal-Programm verbleibt nach Initialisieren eines Windows die Kursorsteuerung im aktuellen Window. Zum Schreiben werden die bekannten Befehle **write();** und **writeln();** benutzt.

Die Window-Box erkennt Steuerzeichen zur Kursorpositionierung, zum Löschen des Bildschirms und zur Helligkeitssteuerung – alle Steuerzeichen werden im aktuellen Window ausgeführt.

Beim Ausschalten der Windowfunktion bzw. beim Initialisieren eines neuen Windows wird der Windowinhalt abgespeichert. Erfolgt ein neuer Aufruf des Windows, wird der alte Inhalt wieder dargestellt, und die Kursorsteuerung wird an der alten Stelle fortgesetzt.

VEB Optima Aschersleben, Abt. F/E, Wilhelmstraße 21/31, Aschersleben, 4320; Tel 5151 App. 238 *Kaul*

Fakten-Recherchesystem

Aufgrund des großen Zuspruchs, den das Fakten-Recherchesystem REDAFAKT für 8-Bit-Mikrorechentechnik gefunden hat, erfolgte nunmehr eine Anpassung an die 16-Bit-Version SCP 1700.

Das dialogorientierte REDAFAKT basiert auf dem relationalen Datenbankbetriebssystem REDABAS und dient der Erfassung, Pflege, Recherche und wahlfreien Ausgabe von Daten fester Länge auf Bildschirm und/oder Drucker. Dem Charakter der zu erfassenden Objekte entsprechend bildet das Grundschema dieses Programmes eine Matrix, die den Aufbau von Dateien mit einer vom Anwender bestimmbaren (beliebigen) Anzahl von Feldnamen gestattet, wobei die maximale Länge eines Feldnamens 19 Zeichen betragen kann. Entsprechend dem Erfassungsmodus erfolgt die Dateneingabe auf Maskenbasis nach dem Prinzip „Feldname: Feldaussage(n)". Die Pflege beinhaltet sämtliche gespeicherten Daten und sichert neben einer permanenten und unkomplizierten Aktualisierung des Datenbestandes die Fortschreibung bis zum physischen Ende einer Diskette. Das Rechercherepertoire umfaßt die Funktionen der Booleschen Algebra und realisiert neben diesen Möglichkeiten die vergleichende und partielle Suche im Datenfonds. Die Dialogsteuerung mittels Menütechnik erfordert bezüglich ihrer gerätetechnischen Bedienung keine datenverarbeitungsspezifischen Kenntnisse und unterstützt eine redundanzarme, Dialogschritte überspringende und/oder zusammenfassende Korrespondenz. Das Programm ist überall dort nutzbar, wo sich der zu dokumentierende Sachverhalt in das o.g. Erfassungsschema eingliedern läßt, der Datenbestand sich qualitativ für die dezentrale Rechentechnik mit ihren spezifischen Speichermedien eignet und tabellarische Aufbereitungsformen zweckmäßig erscheinen.

VEB Robotron-Elektronik Dresden, Leitstelle für Information/Dokumentation, PSF 330, Dresden, 8012; Tel. 4872301
Brünner



Entwicklungen und Tendenzen

MC-68030-Prozessor mit 33 MHz

Motorola hat seine 32-Bit-Mikroprozessorfamilie um eine Version des MC 68030 mit 33 MHz erweitert. Damit dürfte dies der zur Zeit schnellste Mikroprozessor für universelle Anwendungen auf dem Markt sein.
Als einer der ersten Hersteller hat die Firma Apollo Domain bereits Workstations auf Basis dieses MC 68030 vorgestellt. Die neue Serie DS 4500, die es mit Interleaved Memory auf 8 MIPS bringt, nutzt außerdem den mathematischen Koprozessor MC 68882 und einen 64-KByte-Cache. Hewlett-Packard kündigte an, daß der Prozessor in einem Spitzenmodell der Workstation-Serie 9000 eingesetzt werden wird.

MP

16-Bit-Spezialist für Forth

Durch direkte Ausführung von Forth-Programmcode kann der von Harris Semiconductor vorgestellte 16-Bit-Mikrocontroller RTX-2000 über 10 MIPS Befehlsdurchsatz erreichen, weshalb der Baustein für zeitkritische Echtzeit-Steuerungsaufgaben gut geeignet ist. Die denkbaren Einsatzbereiche fangen bei Robotersteuerungen an und umfassen auch komplexe Bildverarbeitungs- und Signalverarbeitungs-Systeme sowie Applikationen der künstlichen Intelligenz.
Der Chip stellt das erste Miglied der von Harris vorgestellten „Real Time Express"-Familie dar, deren Konzept den Anwender von der Notwendigkeit befreit, Programme für Echtzeit-Anwendungen unmittelbar in Assembler zu formulieren. Damit ist diese Controller-Serie die erste, die unmittelbar eine Hochsprache akzeptiert. Die Bausteine verfügen auch über flexible Peripherie-Anschlußmöglichkeiten, so daß die Einbindung in bestehende oder zu konzipierende Systemarchitekturen keine Probleme aufwerfen dürfte.

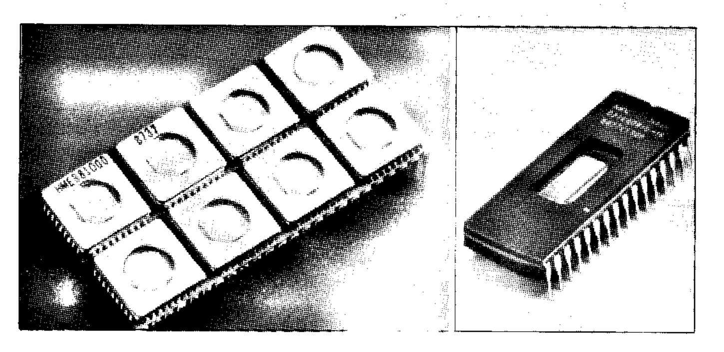
MP

Ventura Publisher Version 2.0

Auch professionelles „Publizieren am Schreibtisch" soll jetzt die neue Version des DTP-Programms Ventura Publisher von Rank Xerox ermöglichen. Lauffähig ist das Programm auf allen IBM PC XT/AT, Kompatiblen und auf dem PS/2. Das „neue Ventura" erlaubt die vollständige Integration von Texten, Grafiken und über Scanner eingelesenen Vorlagen und wird jetzt in drei Versionen angeboten: einer Basis-Version, einer Netz-Version und einer Profi-Erweiterungsversion.
Die neue Version 2.0 weist mehr als 70 Neuerungen gegenüber der Version 1.2 auf. Besondere Merkmale sind: Netzwerkfähigkeit, die Erstellung von Tabellen und Formeln, verbesserte Halbtonverarbeitung, erweiterte Grafikfunktionen, interaktives Kerning, vertikaler Keil, Textrotationen sowie die Erweiterung der Drukker- und Bildschirmtreiber plus Fonts.

MP

EPROMs mit 4 und 8 MBit



Der **HME 881000** ist ein 8-MBit-Hybrid-EPROM (linkes Bild), der aus acht 1-MBit-EPROMs aufgebaut ist. Die Adreß- und Datenleitungen sind mit Decodern und Treibern versehen, so daß sich der gesamte Schaltkreis wie ein 8-MBit-EPROM verhält. Das Gehäuse hat 48 Pins und eine Größe von $61 \times 30{,}5 \times 8{,}6\,mm^3$. Die Zugriffszeiten liegen zwischen 200 und 350 ns.
Das rechte Bild stellt den 4-MBit-EPROM **μPD27C4001DZ** von Nippon Electric (NEC) dar, der ab Januar produziert werden soll. Er wird in 1-μm-CMOS-Technologie gefertigt und mit Zugriffszeiten von 120 bis 200 ns angeboten. In Vorbereitung ist die fensterlose Version dieses Schaltkreises. Beide sind kompatibel zum ROM μPD23C4001E von Kawasaki, der bereits verfügbar ist. Mit einem entsprechenden Programmieralgorithmus kann der μPD27C4001DZ mit 100-ns-Impulsen in 60 Sekunden programmiert werden. Verkappt ist er mit einem 32poligen Keramikgehäuse, das ein großes Fenster für das $5{,}48 \times 14{,}79\text{-}mm^2$-Chip enthält.

MP

Schneller Transputer-Chip

Inmos International hat eine 25-MHz-Version des T800-Transputer-Chips vorgestellt. Die Serienproduktion soll in diesem Jahr anlaufen. Der T800 kombiniert einen 32-Bit-Integer-Prozessor mit einer 64-Bit-Arithmetik-Einheit. Inmos gibt die Leistung des 25-MHz-Chips mit 2,9 MFLOPS an. Eine 30-MHz-Version ist geplant.

MP

Superminicomputer MV 40000

Die Firma Data General stellte kürzlich ihre 32-Bit-Superminicomputerfamilie MV 40000 vor, die sie als ihr bisher leistungsfähigstes System bezeichnet. Das Modell MV/40000 beispielsweise ist kompatibel zu den bisherigen MV-Produkten und erlaubt die Verarbeitung von bis zu 14 MIPS (Millionen Operationen pro Sekunde). Noch leistungsfähiger ist das Modell 40000 HA (High Availibility), das bis zu vier symmetrische Multiprozessoren unterstützt und es damit bis auf 50 MIPS bringt. Damit soll es etwa die doppelte Leistung der Vierprozessor-VAX 8840 haben, die von DEC als Spitzenmodell eingestuft wird.

MP

Kombination von Silizium- und Galliumarsenid-Technik

In der Mikroelektronik ist Silizium heute der Hauptwerkstoff. Die meisten integrierten Schaltungen, die in Computern und anderen Elektronikprodukten zu finden sind, bestehen aus feinsten, in das Silizium eingebrachten Strukturen. Aber auch der Halbleiter Galliumarsenid findet wegen der damit möglichen hohen Schaltgeschwindigkeiten für spezielle Zwecke Anwendung. Die Nachteile dieses Materials begrenzen seinen Einsatz. Arsen ist giftig, und die Herstellung von Wafern (Scheiben, aus denen Chips gefertigt werden) aus Galliumarsenid ist schwieriger und teurer als aus Silizium.
Neue Möglichkeiten ergeben sich jetzt mit einer Technik, die es gestattet, Galliumarsenid auf eine Siliziumunterlage aufzubringen. Damit gelingt es, die Vorteile beider Halbleiter – Geschwindigkeit und bewährtes Herstellungsverfahren mit großer Integrationsdichte – auf einem Chip zu vereinen. Geplant sind Chips, die auf einem Baustein gleichzeitig preiswerte, aber vergleichsweise langsame Silizium-Transistoren und die schnelleren Galliumarsenid-Elemente tragen. Da diese nicht so dicht angeordnet werden können wie Siliziumtransistoren, übernehmen sie bei einem derartigen Schaltkreis nur die Aufgaben, für die höchstes Tempo erforderlich ist.
Die schnellere Schaltgeschwindigkeit in Galliumarsenid beruht darauf, daß sich in diesem Halbleiter die Elektronen etwa fünfmal so schnell bewegen wie in Silizium. Wegen dieser höheren Beweglichkeit genügen geringere Spannungen zum Beschleunigen, die Leistungsaufnahme der Bauelemente ist dadurch kleiner. Allerdings ist das Aufbringen von Galliumarsenid auf Silizium sehr kompliziert. Zwar bestehen beide Stoffe aus einfachen kubischen Kristallgittern, jedoch mit unterschiedlichen Atomabständen, so daß die Gitter nicht genau aufeinander „passen".
Neben der Herstellung kombinierter Silizium-Galliumarsenid-Chips gilt derzeit auch der Produktion reiner Galliumarsenid-Schaltungen auf Siliziumwafern das Interesse der Forscher. Ziel ist es, kostengünstig leistungsfähigere Chips fertigen zu können. Ein Teil der komplizierten Schaltkreis-Herstellung soll nach der bekannten und bewährten Siliziumtechnik ablaufen; nur für wenige Produktionsschritte ist die schwierigere Galliumarsenid-Technik vorgesehen.
Mit einer gemischten Galliumarsenid- und Siliziumtechnik ist es amerikanischen Forschern bereits gelungen, einen statischen Schreib-Lese-Speicher (SRAM) zu fertigen. Er enthält 7500 Transistorfunktionen. Ein weiterer Baustein, der verschiedene logische Operationen bewältigt, umfaßt 6000 Transistoren. Wie überlegen dem derzeit noch die Herstellungsverfahren für reine Siliziumchips sind, wird daran deutlich, daß dort auf einer wenige Quadratmillimeter großen Fläche viele Millionen Bauteilfunktionen untergebracht werden können.

ADN

Betriebssysteme in Konkurrenz

Das Marktforschungsunternehmen Diebold erwartet, daß das Mikrocomputer-Betriebssystem MS/DOS noch etwa vier Jahre seine Marktstellung halten wird. Ab 1993, so die Managementberater, werde sich OS/2 durchsetzen. Auf der Ebene der Zentralrechner sollen herstellerspezifische Betriebssysteme ihre hohe Marktbedeutung behalten. Das gilt vor allem für die IBM-Betriebssysteme MVS, VM und auch noch VSE, ebenso für das Siemens-System BS 2000.
Im Bereich der mittelgroßen Rechner werden nach Überzeugung Diebolds herstellerspezifische Betriebssysteme wie VMS (Digital Equipment), VSE (IBM), MPE (HP) oder OS/400 (IBM) in harter Konkurrenz mit dem Standardbetriebssystem Unix stehen, das künftig wachsende Akzeptanz finde, auch wenn seine Funktionalität noch nicht das hohe Niveau herstellerspezifischer Betriebssysteme erreiche. Auf der Workstation-Ebene hätte Unix 1987 bereits einen Absatzanteil von 55 bis 65 Prozent gehabt. Im Preis-/Leistungsverhältnis weise Unix Vorteile gegenüber der herstellerspezifischen Konkurrenz auf.

MP

Neuer Supraleiter

Die Entdeckung eines dritten Typs der neuen Hochtemperatursupraleiter meldete kürzlich ein Forscherteam des Nationalen Forschungsinstituts für Metalle der japanischen Agentur für Wissenschaft und Technik. Dieses Material enthält keine seltenen Erden und setzt sich aus Wismut, Strontium, Kalzium und Kupferoxid zusammen. Es läßt sich relativ einfach herstellen und weist zwei supraleitende Phasen auf, wobei die Phasen mit der höheren Temperatur bei 105 Kelvin (Minus 168 Grad Celsius) supraleitend wird. Seine Struktur scheint sich völlig von der herkömmlicher Supraleiter zu unterscheiden. Umfassende Analysen sollen helfen, neue Erkenntnisse über die Mechanismen der Supraleitung zu gewinnen.

ADN

Schritte zur Realisierung neuraler Rechnernetze

Informationsverarbeitungssysteme, die in Analogie zu denen der Lebewesen arbeiten, gehören zu den am schnellsten expandierenden Forschungsbereichen.
Seit 2 Jahren werden neurale Netzwerke kommerziell genutzt. Gegenwärtig sollen 150 solcher Netzwerke im Einsatz sein. Die aus einer Vielzahl gekoppelter Prozessoren bestehenden Netze verfügen über eine wesentlich höhere Verarbeitungsleistung als konventionelle Rechner. Äußerst aufwendig ist der Entwurf derartiger Netze. Bisher wurden die Netze auf konventionellen Rechnern simuliert. Die Simulation eines parallelen Systemkonzeptes auf einem konventionellen Rechner erfordert jedoch sehr hohe Rechenzeiten. Die Firma Texas Instruments soll ein Rechnersystem entwickelt haben, das die effektive Simulation neuraler Netze gestattet. Die Verarbeitungsleistung dieses Rechners entspricht der von 100 Minicomputersystemen. Grundlage des Computers mit der Bezeichnung *Griffin* sind ein spezifischer Chip zur Simulation und ein Kommunikationschip. Diese Chips sind mit einem konventionellen Mikroprozessor und dem Hauptspeicher auf einer Leiterplatte (Netsim-Karte) verbunden. Der Rechner besteht aus 36 Leiterplatten. Der Kommunikationschip ermöglicht zwischen der Karte einen Informationsaustausch mit 10 Megabit/Sekunde. Die einzelnen Leiterplatten, die Minicomputersysteme darstellen, sind mit einem konventionellen Personalcomputer als Hostrechner gekoppelt. Der Hostrechner übernimmt die Korrespondenz mit dem Nutzer, die Herstellung der Anfangszustände, die Realisierung der Eingaben und Ausgaben sowie die Fehlerbeseitigung.
Diese Kombination des Personalcomputers mit den Netsim-Karten vereint einen einfachen Zugang zum Rechner und die Nutzung hoher Verarbeitungsgeschwindigkeiten.

Quelle: Blick durch die Wirtschaft vom 22. 8. 1988 *Wi*

Bildverarbeitung mit Video-Geschwindigkeit

Die japanische Firma Fujitsu hat nach eigenen Angaben mit dem Verkauf des international ersten Bildverarbeitungsverfahrens begonnen. Dieses Bildverfahren verarbeitet bewegte Farbbilder, die mit Fernsehkameras aufgenommen wurden, in Video-Geschwindigkeit. Bei dem neuen Verfahren mit der Bezeichnung *Fujitsu Integrated Visual Information System/Video-Rate Image Processing System* (FIVIS/VIP) wird eine variabel strukturierbare Leistungsarchitektur verwendet, um bewegte Bilder in Video-Geschwindigkeit von 60 Aufnahmen pro Sekunde zu analysieren, zu verarbeiten und evtl. zu verändern oder zu ergänzen.
Das neue Verfahren nutzt zehn Hochgeschwindigkeits-Bildverarbeitungs-Basismodule, die gleichzeitig und parallel zueinander arbeiten. Sie erkennen Farben, entdecken sich bewegende Objekte und Formen und können die Position von Gegenständen ausmachen. Das neue Verfahren könnte in zahlreichen Gebieten, wie bei der Kontrolle von Erzeugnissen im Fertigungsprozeß und beim Objektschutz, Anwendung finden.
Wird das Verfahren in Verbindung mit Infrarot-Detektoren oder multiplen Kameras zum Einsatz gebracht, könnte es auch zum Aufspüren von Unregelmäßigkeiten in der Hitzeverteilung oder zur Entfernungsmessung in Echtzeit verwendet werden.
Für das neue Verfahren setzt Fujitsu einen Minicomputer Facom A-50S ein.

Fa

LCD-Farbbildschirm von Sharp mit 1,2 Mio Bildpunkten

Die Firma Sharp entwickelte einen neuen hochauflösenden Farbbildschirm in LCD-Technik mit 1,2 Mio Bildpunkten. Der Bildschirm, der 2,7 cm dick, 27,8 cm breit und 22,1 cm hoch ist, wiegt nur 1,8 kg. Das Farbbild besteht aus 308160 in Deltakonfiguration angeordneten Bildelementen, so daß insgesamt 1232640 Bildpunkte vorhanden sind. Probleme bereitet bei solch einer Konstruktion die Ansteuerung, denn es muß jeder Bildpunkt einzeln angeregt werden. Die Firma will dieses Problem mit Hilfe der Dünnfilm-Transistor-Technik lösen. In 17,5 ms baut sich das Bild auf und steht ohne erkennbares Flimmern. Der seitliche Betrachtungswinkel von links nach rechts liegt bei 60 Grad, von oben bei 20 Grad und von unten bei 30 Grad. Der Kontrast wird mit 1 zu 1000 angegeben. Ob dieser Bildschirm auch für Bewegtbilder einsetzbar ist, wird noch getestet.

Quelle: Blick durch die Wirtschaft vom 4. 8. 1988

Fa

Forschungen zur Molekularelektronik in Großbritannien

Im Rahmen ihres Link-Plans unterstützt die britische Regierung fünf neue wissenschaftliche Programme. Eines dieser Programme ist die Erforschung der Molekularelektronik mit dem Ziel, einen völlig neuartigen Computer zu entwickeln. Dabei könnte es sich um einen chemischen oder biochemischen Computer handeln, der organische Moleküle nutzt, um Informationen zu verarbeiten. Die britische Regierung unterstützt dieses Programm mit einer Investition in Höhe von 10 Mio Pfund. An diesem Programm arbeitet eine Gruppe von Wissenschaftlern aus Industrie und Forschung. Diese Arbeitsgruppe hat zunächst versucht, eine Auswahl unterschiedlicher Forschungsrichtlinien aufzustellen. Das Endziel soll ein Computer ohne die Begrenzung der Silizium-Chips sein, bei dem u. a. keine Probleme hinsichtlich des Wärmeverlustes auftreten. Die Wissenschaftler bezeichnen ihren Computer als *Supermolekular-Informationsprozessor* (SIP). Bei der Molekularelektronik werden organische Moleküle durch elektrische und elektromagnetische Signale angeregt. Ein Beispiel für organische Moleküle, die auf elektrische oder Wärmesignale reagieren, sind die Flüssigkristallanzeigen in Uhren und Fernsehkleinstgeräten. Einen neuen Anwendungsbereich für die Molekularelektronik sieht man in der Möglichkeit, künftig in Filmen oder sogar bei dreidimensionalen Matrizen mit Hilfe des Langmuir-Blodgett-Verfahrens organische Moleküle zu organisieren.
Die Arbeitsgruppe ist der Ansicht, daß dies der einzige gegenwärtig bekannte Weg ist, der zu einer Supermolekularbaugruppe führen könnte. Eine derartige Baugruppe könnte sogar aus biologischen Molekülen bestehen.
Die große Auswahl an Molekularstoffen, besonders an organischen Feststoffen, eröffnet nach Ansicht von Experten verschiedene Möglichkeiten für künftige Fortschritte. Anorganische Stoffe bieten langfristig dagegen relativ wenig Möglichkeiten.
Der Trend geht dahin, organische Stoffe zu entwickeln, die genau über diese benötigten Eigenschaften verfügen.
Im Rahmen des britischen Link-Plans wird die Molekularelektronik als „systematische Erschließung molekularer, einschließlich makromolekularer und biomolekularer Stoffe für die Elektronik sowie verwandte Gebiete, wie z. B. die Optoelektronik", definiert.
In den nächsten zehn Jahren sollen Hybridverfahren entwickelt werden, bei denen ein Gerät teils auf herkömmlichen Feststoffen und teils auf Molekularelektronik beruht.
Auf der Basis der Hybridverfahren sollen völlig neue Gerätetypen zum Speichern, Verarbeiten, Übertragen und Umwandeln von Daten, die vollständig aus der Molekularelektronik abgeleitet werden, folgen. Die Wissenschaftler wollen außerdem noch Energiequellen in Form von Batterien mit einer dünnen Schicht als wesentlichem Bestandteil des chemischen Chips herstellen.
Die Arbeitsgruppe untersucht vier Möglichkeiten, wie die Elektronik mit Hilfe von Langmuir-Blodgett-Schichten (L-B-Schichten) weiterentwickelt werden kann. Die erste Möglichkeit bietet sich aufgrund der nichtlinearen Eigenschaften, bedingt durch ihr einzigartiges Herstellungsverfahren. Die zweite Möglichkeit beruht auf mikrolithografischem Fotolack für künftige Chipgenerationen. Tunneleffekt-Abstandsringe sind die dritte Möglichkeit, wobei man die Tatsache ausnutzt, daß ihre Stärke reduziert werden kann bis zu einem Nanometer.
Die vierte Möglichkeit stützt sich auf Sensoren, besonders Biosensoren, die z. B. aus Enzymen oder Antikörpern bestehen. Ein derartiger Sensor könnte in der Lage sein, eine immunologische Reaktion zu steuern. Er könnte sogar den Weg für einen Biocomputer ebnen, bei dem lebende Organismen verwendet werden.

Quelle: Financial Times vom 4. 2. 1988

Wi

Weitere Kapazitätserhöhung bei 3,5-Zoll-Winchesterlaufwerken

Dem Bestreben nach Miniaturisierung der Informationsverarbeitungsgeräte kommen auch die Speicherproduzenten nach. Die Firma Maxtor stellte als Spitzenmodell ein 3,5-Zoll-Winchesterlaufwerk vor, das eine Kapazität von 200 MByte bei einer Zugriffszeit von 10 ms aufweisen soll, wobei eine weitere Steigerung in Aussicht gestellt wurde.
Der Einsatz ist besonders für Workstations, PCs und tragbare Computer vorgesehen. Besonderer Wert wurde deshalb auf Sicherheit unter rauhen Betriebsbedingungen gelegt. So beträgt die Stoßfestigkeit 50 g. Der MTBF-Wert wird mit 50000 Stunden und die Leistungsaufnahme mit 10 W angegeben. Gleichzeitig wurde für den Massenmarkt ein Modell mit einer Speicherkapazität von 100 MByte, einer Zugriffszeit von 30 ms, einem MTBF-Wert von 40000 Stunden und einer Leistungsaufnahme von 13 W angekündigt.

Quelle: Elektronik. – München 37(1988)17. – S. 52

Fa

Additivtechnik modernisiert Leiterplattenfertigung

In Japan wird derzeit bereits ein Drittel aller Leiterplatten, insbesondere die für technisch und qualitativ hochwertige Industrieelektronik, nach einem neuen Herstellungsverfahren gefertigt. Es wird als Additivtechnik bezeichnet und soll der zunehmenden Miniaturisierung gerecht werden, die nicht nur modernste Verfahren für die Bauelementeherstellung, sondern auch für ihre Träger erfordert.
Das erste, jahrzehntealte, aber noch immer am meisten angewandte Verfahren zur Herstellung von Leiterplatten ist die Subtraktionstechnik. Als Grundmaterial dafür dient dünnes, vollständig mit Kupferfolie beschichtetes Isoliermaterial wie Papier oder Glasepoxi. Nach dem Aufbringen einer Fotomaske ätzt man von dieser Schicht das Kupfer so ab, daß nur die Lötaugen und die sie verbindenden Leiterbahnen stehen bleiben. Im Detail erfordert das etliche kostenintensive Arbeitsgänge. Feinste Leiterbahnen, wie sie für die heute gefragten Packungsdichten erforderlich sind, lassen sich damit kaum erreichen.
Diese Nachteile überwindet die Additivtechnik. Bei diesem Verfahren wird Kupfer aus einem chemischen Bad nur an genau den Stellen auf dem Grundmaterial abgeschieden, wo Leiterbahnen sein sollen. Schwierigkeiten bereitete dabei zunächst die Bindung des abgeschiedenen Kupfers auf dem Trägermaterial. Ein „Kleber" wurde gesucht, der eine lötfeste Verbindung garantiert.
Diesen Kleber fanden USA-Wissenschaftler mit ihrem A-50-Verfahren, bei dem das Kupfer aus einem 73 bis 74 Grad Celsius warmen Kupfersulfatbad abgeschieden wird. Die darin enthaltenen Zusätze von Palladium, Chromschwefelsäure und Natriumfluorid sorgen für sichere Haftung.
Die Feinheit der Leiterbahnen betreffend, meinten die Entwickler des Verfahrens, daß hier eher die fotochemischen Vorlagen und Vorgänge beim Aufbringen des Leiterbahnmusters Grenzen setzen als die Additivtechnik, mit der auch schon sogenannte Multilayer-Leiterplatten gefertigt werden, bei denen sich mehrere Lagen von Leiterbahnen übereinander befinden.

ADN

44. Internationale Technische Messe Plowdiw '88

Vom 26. September bis 2. Oktober 1988 fand in Plowdiw (VRB) zum 44. Mal die Technische Herbstmesse mit wiederum gestiegenen Aussteller- und Besucherzahlen statt. Es beteiligten sich 33 Länder mit 3676 Firmen, darunter 2401 ausländische Aussteller. Wie der Generaldirektor der Messe, Kiril Asparuchow, erläuterte, ist diese Steigerung vor allem der Stimulierung des Handels und der Industriekooperation zwischen bulgarischen und ausländischen Firmen zu verdanken. Grundlage dafür ist die auf dem Juliplenum 1987 beschlossene wirtschaftliche Umgestaltung, gekennzeichnet durch Selbstverwaltung der Betriebe, ihre Freizügigkeit, mit bulgarischen und ausländischen Firmen Gemeinschaftsunternehmen zu gründen, die Tendenz zum Technologietransfer mit dem Ziel, die bulgarischen Erzeugnisse konkurrenzfähig zu machen sowie die neugeschaffenen Handelsbanken mit erweiterten Rechten. Für den Messebesucher dokumentierte sich diese Entwicklung in einer Vielzahl von bulgarischen Anbietern zum Beispiel für Computertechnik, wenngleich bei den kleinen Firmen nicht in dem Umfang wie auf den Messen in Budapest und Poznan. Das Angebot umfaßte eine breite Palette, die auch Spitzenprodukte nicht ausließ.

Der Betrieb für Computertechnik Sofia ist Hersteller des Isotimpex-ZIT **Super 32** (Bild 1; Bilder siehe 2. und 3. Umschlagseite), der als voll kompatibel zum IBM PC, XT, AT und zum PS/2 (?) bezeichnet wird. Eingesetzt werden kann er sowohl als Einplatz-PC als auch in Netzwerken. Als CPU wird der 80386 verwendet, dessen Taktfrequenz hard- oder softwaremäßig zwischen 16 und 20 MHz umgeschaltet werden kann. Option ist ein arithmetischer Koprozessor. Der Hauptspeicher umfaßt 2 MByte auf der Mutterplatine, wobei RAMs mit 100 ns Zugriffszeit und wahlweise mit 80 ns Zugriffszeit verwendet werden. Der gesamte RAM kann aufgeteilt werden in den Basisspeicher mit 640 KByte, den Expansions-RAM mit 1024 KByte und ein Shadow-RAM mit 384 KByte (in den Shadow-RAM kann z. B. das ROM-BIOS geladen werden, um kürzere Zugriffszeiten zu erreichen).

Als externe Massenspeicher stehen 40-MByte-Festplatten (wahlweise 80 MByte und mehr), 5,25-Zoll-Disketten mit 1,2 MByte und 3,5-Zoll-Disketten mit 720 KByte in Slim-line-Ausführung sowie ein 60-MByte-Bandlaufwerk zur Auswahl. Der Super 32 arbeitet mit der Farbgrafik EGA, für die der spezielle 14-Zoll-Monitor geliefert wird. Zur Auswahl wird ein monochromer Monitor und ein hochauflösender 14-Zoll-Farbmonitor angeboten. Acht freie Leiterkartensteckplätze können über den Expansionsbus wie folgt genutzt werden:

2 Stück für PC-kompatible 8-Bit-Erweiterungskarten, 5 Stück für AT-kompatible 16-Bit-Erweiterungskarten, 1 Stück für die 32-Bit-RAM-Erweiterungskarte.

Als Betriebssysteme können MS-DOS und XENIX genutzt werden; Compiler gibt es für Fortran, Pascal, C und GWBasic. Die Anwendungssoftware enthält unter anderem P-CAD und ACAD der Autodesk Inc.

Während der Super 32 als Tower angeboten wird, der Platz für die zahlreichen Erweiterungsleiterkarten bietet, produziert das Kombinat für Mikroprozessorsysteme Prawez ihren ersten 32-Bit-PC, den **Pravez-386** (Bild 2), als Desk-Top-, also Tischmodell. Basis ist der Prozessor 80386, der um einen 80287- bzw. 80387-Koprozessor ergänzt werden kann. Der 1-MByte-RAM kann bis zu 10 MByte erweitert werden, der ROM hat 128 KByte. Als externe Speicher gibt es 5,25-Zoll-Disketten mit 1,2 MByte (bzw. 360 KByte) sowie Festplatten mit 10 oder 20 MByte.

Auch der Pravez-386 kann als Einzel-PC wie als Server in lokalen Netzen mit Ring- oder Linienstruktur verwendet werden. Die Farbgrafik des 386 ist sowohl zum Standardgrafikadapter CGA als auch zur höherauflösenden EGA-Grafik kompatibel. Die Leistungsfähigkeit des 80386-PCs erlaubt bereits umfangreiche grafische Anwendungen unter Nutzung von bis zu 8 Terminals. So wurden mittels des Pravez-386 die Leistungen des auch angebotenen CAD-Paketes **TIGS-R** (Bild 3) demonstriert. Das Programm ist kompatibel zu AutoCAD und bietet die bekannten Möglichkeiten der Zeichnungserstellung mit Dimensionierung und der grafischen Ein- und Ausgabe. Voraussetzung für die Anwendung von TIGS-R ist ein PC-DOS-kompatibles Betriebssystem; beispielsweise das PDOS des Pravez-386. Als Multitasking-multiuser-Betriebssystem steht für den Computer darüber hinaus noch das XENIX-kompatible INMOS bereit.

Vor etwa 10 Jahren wurde die zum bulgarischen Sportbund gehörende Firma Infosport gegründet, die seit etwa 4 Jahren selbst Hardware produziert und sie einschließlich Software für den Sport-Bereich inzwischen auch zum Verkauf anbietet. So konnten Produkte bereits in Österrich, in der Schweiz, im Iran und Irak abgesetzt werden. Die Zusammenarbeit mit ausländischen Firmen soll künftig verstärkt werden und sich nicht auf den Sportsektor beschränken. Beispielsweise ist Infosport Distributor von Microsoft in 17 Ländern; MS-Windows wird als Standard in kyrillischer Schrift und mit der Firma Ensch ein DTP-System entwickelt.

Für den **Sport 286 Quatro** (Bild 4, rechts) wurde als Basisplatine eine spezielle Leiterkarte entwickelt, die bereits bis zu 8 MByte RAM aufnehmen kann. Damit werden nicht nur Steckplätze für zusätzliche Leiterkarten eingespart, sondern durch Verwenden schnellerer Chips wird auch die Verarbeitung beschleunigt. Der 16-Bit-80286-Prozessor kann hardware- und softwaremäßig auf 8, 10, 16 oder 20 MHz eingestellt werden. Die zusätzliche Verwendung des Koprozessors 80287 ist möglich. Vom Aussteller wurde hervorgehoben, daß das System nach dem Landmark-Speedtest eine Geschwindigkeit von 26,7 MHz erreicht (dabei ist jedoch zu beachten, daß dieses Testverfahren in keiner bzw. nur geringer Weise langsame Peripherie und langsame Speicher berücksichtigt; es ist damit für den praktischen Gebrauch nur von untergeordneter Bedeutung).

In der Grundausstattung enthält der Sport 286 Q einen EGA-Monitor, optional ist VGA-Farbgrafik lieferbar; als externe Speicher sind 5,25-Zoll-Floppy mit 1,2 MByte und 80-MByte-Festplatte vorhanden (wahlweise 2 Festplatten mit 80, 115, 160 oder 380 MByte). Die angebotene Software umfaßt u. a. MS-DOS 3.3 und OS/2, MS Windows 2.03 und MS Mouse.

Als leistungsstärkstes Modell offerierte Infosport auch einen PC mit dem 32-Bit-Prozessor 80386, der wahlweise mit 16, 20 oder 25 MHz betrieben werden kann. Nach dem bereits erwähnten Landmark-Test sollen sich dementsprechend Leistungen von 22, 27 bzw. 34 MHz ergeben. Als Gesamtleistung des **Sport 386** (Bild 4, links) wird mehr als das 18fache eines PC XT genannt.

Der Hauptspeicher kann auf der Basisplatine von 2 auf 8 MByte aufgerüstet werden; für die Erweiterung bis zu 16 MByte steht ein 32-Bit-Slot zur Verfügung. Als weitere Speicher dienen ein Diskettenlaufwerk mit 5,25 Zoll und 1,2 MByte, ein 3,5-Zoll-Laufwerk mit 720 KByte oder 1,44 MByte, eine Festplatte mit 80 MByte und ein Streamer mit 60 MByte. Als Option gibt es die Variante mit 2 Harddisks mit je 115, 160, 315 oder 382 MByte Kapazität. In der Grundausstattung ist das System mit einem EGA-Monitor ausgestattet, optional mit VGA. Auch zum Sport 386 gibt es MS-DOS 3.3, OS/2, MS-Windows/386, MS-Mouse sowie Anwendungs- und Netzsoftware.

Das Institut für Technische Kybernetik und Robotertechnik der bulgarischen AdW stellte in einem gesonderten Bereich eine Vielzahl von Entwicklungen vor:

Mit dem Bildverarbeitungssystem **CSI-11** (Bild 5) wurde eine preiswerte Lösung angestrebt, die es erlaubt, in Verbindung mit einem XT-kompatiblen Personalcomputer (im Bild Pravez EC 1839) über eine Videokamera aufgezeichnete Bilder auszuwerten und zu bearbeiten. Der eingebaute MP-Modul mit MC 6809 und 32 KByte RAM/EPROM erlaubt jedoch auch die autonome Arbeit ohne PC. Weitere Bestandteile sind: Synchronisationsblock, ADU mit 6 Bit Auflösung und 6,25 MHz Umsetzrate, asynchrone RS-232-C-Schnittstelle, Parallelinterface, DMA für die Kopplung mit dem PC sowie Videospeicher mit 256 KByte. Er ermöglicht die Aufnahme von 4 Bildern mit 256 × 256 × 8 Bit, wobei die Übertragung der Informationen von einer Videoseite zur anderen und die Verarbeitung in Echtzeit möglich sind.

Die Haupteinsatzgebiete liegen damit in der Kontrolle industrieller Prozesse sowie im Einsatz für „sehende" Roboter, medizinische Untersuchungen und Sicherungssysteme.

Als abschließendes Produkt aus dem Institut sei das Desktop-Publishing-Programm **RedPak** genannt, welches auf einer PC-Minimalkonfiguration vorgeführt wurde (Bild 6). Es ist vergleichbar mit dem bekannten Layoutprogramm Pagemaker und erlaubt, mit dem PC erstellte Textfiles, Auto-CAD-Daten, Paint-Brush-Grafiken und gescannte Bilder auf dem Bildschirm zu bearbeiten; das heißt, Text und Bilder gleichzeitig für die Ausgabe einer Druckseite aufzubereiten. Von dem Entwickler kann nicht nur das Layoutprogramm, sondern auch eine DTP-Hardware-Konfiguration bezogen werden. Zum Beispiel: PC AT mit 1 MByte RAM, 40-MByte-Festplatte, 1,2-MByte-Floppy, 360-KByte-Floppy, Monochrom-Monitor mit 1024 × 780 Pixel, HP-Scanjet-Scanner und HP-Series II-Laserdrucker, wofür etwa 12000 US-$ zu veranschlagen sind.

Die auf diesem System zu verarbeitenden Bild- bzw. Schriftelemente können mit einem ebenfalls entwickelten Programmsystem **R&V** (Raster- und Vektor-Fonts) generiert werden (Bild 7), wofür sich auch ein Scanner als Eingabegerät nutzen läßt.

Vor vier Jahren wurde mit etwa 300 Mitarbeitern der volkseigene Betrieb Softwareprodukte und Systeme (SPS) gegründet, der heute etwa 1500 Mitarbeiter beschäftigt. Zielsetzung des Betriebes ist es, vor allem durch Einbeziehung anderer Partnerbetriebe, also durch Kooperation, von Kunden benötigte spezifische Hard- und Softwarelösungen zusammenzustellen und die notwendigen Anpassungsarbeiten auszuführen. Das Messeangebot umfaßte insgesamt 181 Exponate, darunter 172 Programmlösungen. Die Palette reichte von Anwendungssoftware für die verschiedensten volkswirtschaftlichen Bereiche bis zu anspruchsvollen Satzsystemen mit der kompletten Hard- und Software. Mit dem Kombinat Robotron wurde zum Beispiel gemeinsam das Kontrollsystem **ROMI** entwickelt (im Bild 8 rechts vorn), dessen Grundlage die Datenstation robotron K 8915 ist. Das Betriebsdatenerfassungssystem ermöglicht u. a. die Anwesenheits-, Zutritts- und Arbeitszeitkontrolle.

Abschließend einige Bemerkungen zu neuen bulgarischen peripheren Geräten:

In Kopplung mit einem PC wurde ein Scanner gezeigt, der **CC-25** (Bild 9), welcher mit einer Auflösung von 300 dpi (dots per inch, Punkte pro Zoll) dem heute im DTP-Bereich international vorhandenen Leistungsniveau entspricht.

Ebenfalls in Konfiguration mit einem PC-kompatiblen Computer arbeitet die **Disketten-Duplizierstation** (Bild 10) für 5,25-Zoll-Disketten.

Isot stellte innerhalb des umfangreichen Angebotes an Speichertechnik das neue 5,25-Zoll-Floppy-Laufwerk **EC 5327.01** mit 1,6 MByte (Bild 11) und das Festplattenlaufwerk **CM 5509** mit 31 MByte vor (Bild 11).

Text und Fotos: Hans Weiß



6

6 *Layoutprogramm RedPak*

7 *Generierung der Fonts für RedPak*

8 *BDE-System ROMI*

9 *Scanner CC-25*

10 *Diskettenduplizierstation*

11 *Floppylaufwerk EC 5327.01*

12 *Festplattenlaufwerk CM 5509*

7

8

9

10

11

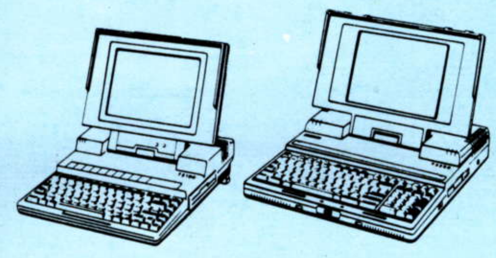
12

ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 1
S. 1–32

Technik international



Laptops


T 5100 T 5200

Zunehmender Beliebtheit erfreuen sich auf dem internationalen Markt die transportablen Personalcomputer. Den größten Anteil dürften aus dieser Gruppe die sogenannten Laptops (Schoßcomputer) haben. Sie liegen mit ihrer Masse etwa um die 6 kg und haben fast alle den gleichen grundlegenden Aufbau (siehe auch MP 9/88, 4. Umschlagseite). Das heißt, die Grundfläche beträgt etwa 30 cm × 35 cm, und das Display ist aufklappbar.
Die Monitore unterscheiden sich jedoch in Aufbau, Format und Handhabung. Verwendet werden vor allem Flüssigkristall- (LC-) und Plasmadisplays. Erstere haben den Vorteil des geringeren Energieverbrauchs (bei Akkubetrieb wichtig), sind jedoch vergleichsweise träge, empfindlich gegenüber geringen Temperaturen und nicht sehr kontrastreich. Eine Verbesserung hat hier die sogenannte Supertwisttechnik gebracht, die auch den möglichen Blickwinkel vergrößert. Oft werden jetzt die Displays mit Hintergrundbeleuchtung (Backlight) ausgestattet. Selbstleuchtende Gas-Plasmadisplays sind den LCD in vielem überlegen, haben aber einen recht hohen Stromverbrauch. Bei starkem Umgebungslicht ist das Ablesen oft problematisch. Elektrolumineszenztechnik wird nur selten verwendet.
Bei Laptops gibt es bis jetzt noch keine Farbdisplays. Zwar verfügen fast alle angebotenen Geräte über einen der gängigen Farbgrafikadapter – CGA (640 × 200 Pixel), EGA (640 × 350 Pixel), selten Olivetti-Modus (640 × 400 Pixel), neuerdings sogar den von IBM für das PS/2 entwickelten VGA-Standard (640 × 480 Pixel) –, sie setzen die potentiellen Farben jedoch für den eigenen Monitor in Grauwerte um. Die meisten Laptops bieten allerdings einen RGB-Ausgang, um externe Farbmonitore anschließen zu können.
Immer häufiger werden Laptop-Computer mit eingebautem Akku für netzunabhängigen Betrieb angeboten. Hersteller dieser Geräte betrachten dies sogar als ein entscheidendes Merkmal, einen Laptop als solchen bezeichnen zu dürfen. Hier gehen die Meinungen jedoch auseinander.
Obwohl man sich vorstellen kann, daß Laptops extrem kompakt aufgebaut sein müssen, um die Leistung eines Personalcomputers in das Aktentaschenformat bringen zu können, warten die meisten Hersteller noch mit freien Leiterkarten-Steckplätzen auf, beispielsweise um Modems oder Netzwerkkarten verwenden zu können. In der Grundausstattung beinhalten Laptops bereits ein 3,5-Zoll-Diskettenlaufwerk (1,44 MByte) und meist eine Festplatte zwischen 20 und 100 MByte. Weitere Speicher sind extern nutzbar. Als Prozessor werden durchweg Intel- oder intelkompatible Typen verwendet, oft in der stromsparenden CMOS-Bauart. Damit ist die Möglichkeit der Arbeit unter MS-DOS mit entsprechender Standardsoftware gegeben. Je nach Einsatzzweck werden die 16-Bit-Typen 8088/8086 oder 80286 bzw. 80C286 verwendet.
Eine Vorreiterrolle übernahm im Herbst 1987 die japanische Firma Toshiba mit der Vorstellung eines Laptops mit dem 32-Bit-Prozessor 80386. Der **Toshiba T 5100**, den unser Farbbild zeigt, besitzt neben einem 3,5-Zoll-Diskettenlaufwerk (umschaltbar von 720 KByte auf 1,44 MByte) eine stoßgeschützte 40-MByte-Festplatte mit 29 ms Zugriffszeit und einem Interleave-Faktor 1.
Der Prozessor wird mit 16 MHz getaktet und benötigt 1 Wait State. Der 2-MByte-RAM kann auf 4 MByte erweitert werden. Der T 5100 wiegt 6,8 kg. Neben dem Betriebssystem MS-DOS 3.2 ist seit Mitte 1988 auch das T/PIX AT&T Unix V.3 verfügbar, so daß ein T 5100 über die RS 232-Schnittstelle bis zu vier Arbeitsplätze unterstützen kann. Auf dem Gas-Plasmadisplay mit 640 × 400 Pixeln lassen sich vier Graustufen darstellen. Auch das Display dürfte mit dafür verantwortlich sein, daß ein Akkubetrieb nicht effektiv wäre und nur der Einsatz über einen Netzanschluß bleibt.
Anfang 1988 präsentierte jedoch GRID Systems Corp. den ersten netzunabhängigen 80386-Laptop, den **Grid-Case 1530**. Er besitzt die stromsparende CMOS-Prozessorversion 80C386, die mit 12,5 MHz oder mit 6 MHz Taktfrequenz arbeiten kann. Auch dieser Laptop hat ein Gas-Plasmadisplay mit 640 × 400 Pixeln, so daß neben CGA optional auch VGA möglich ist. Bemerkenswert ist auch das geringe Gewicht von nur knapp 5,4 kg.
Ebenfalls einen akkubetriebenen Laptop, und zwar mit dem normalen 80386-Prozessor, stellte Mitte 1988 die Firma Zenith vor. Auch dieser arbeitet aus Gründen des Stromverbrauchs unter der Nennfrequenz des 80386 (16 MHz) mit nur 12 oder 6 MHz (umschaltbar). Das Novum des **Turbosport 386** ist das neuartige LC-Display, als Page-white-Monitor bezeichnet, das schwarze Zeichen auf weißem Hintergrund darstellt (alle anderen Hersteller verwendeten farbigen Hintergrund). Damit wird ein bei diesen Geräten bisher nicht gekanntes Kontrastverhältnis von 1:20 erreicht.
Als Spitzenmodell bei Laptops dürfte gegenwärtig Toshibas neuestes Modell, der **T 5200**, einzustufen sein. Er verwendet den 80386 mit 20 MHz, hat 2 bis 8 MByte RAM, 40- oder 100-MByte-Festplatte mit 25 ms Zugriffszeit und ein 3,5-Zoll-Diskettenlaufwerk. Als Bildschirm wird wie beim Vorgänger T 5100 ein Gas-Plasmadisplay verwendet; mit 640 × 480 Pixeln und VGA-Standard. Auch ist hier zum Betrieb die Steckdose Voraussetzung. In den Maßen (379 × 395 × 99 mm^3) und im Gewicht (8,5 kg) übertrifft er jenen dagegen, wie auch unsere grafische Gegenüberstellung zeigt.
Mittlerweile haben fast alle namhaften Hersteller von Personalcomputern Laptops in ihrem Programm.
So auch seit Okober 1988 Compaq mit dem **SLT/286**, den wir in der Rubrik Entwicklungen und Tendenzen in MP 12/88 bereits vorstellten. Mit diesem Laptop wird erstmals ein Gerät angeboten, das den VGA-Grafikmodus auf einem LC-Display realisiert.
Auch weiterhin ist mit einer stürmischen Entwicklung bei dieser Computerkategorie zu rechnen. Gleichermaßen beim Absatz der Geräte, der zwar – sicher wegen des im Vergleich zu Tischmodellen höheren Preises – nicht so hoch wie ursprünglich vorausgesagt, aber dennoch deutlich steigend ist.

MP-We

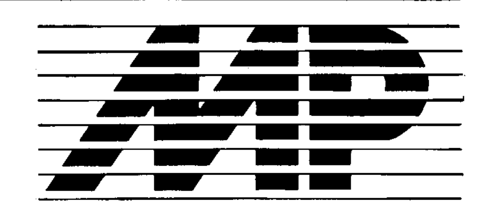
Foto: Weiß; Grafik: Toshiba



ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 2 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
Diskettenformate
in CP/M-Systemen
DISKETTEN
FDC U 8272 D
in der Anwendung
8 × 8 –
ein Font für alle Fälle

.0 D0H	.1 D1H	.2 D2H	.3 D3H	.4 D4H	.5 D5H	.6 D6H	.7 D7H	.8 D8H	.9 D9H	.10 DAH	.11 DBH
.12 DCH	.13 DDH	.14 DEH	.15 DFH	.16 10H	.17 11H	.18 12H	.19 13H	.20 14H	.21 15H	.22 16H	.23 17H
.24 18H	.25 19H	.26 1AH	.27 1BH	.28 1CH	.29 1DH	.30 1EH	.31 1FH	.32 20H	.33 21H	.34 22H	.35 23H
.36 24H	.37 25H	.38 26H	.39 27H	.40 28H	.41 29H	.42 2AH	.43 2BH	.44 2CH	.45 2DH	.46 2EH	.47 2FH
.48 30H	.49 31H	.50 32H	.51 33H	.52 34H	.53 35H	.54 36H	.55 37H	.56 38H	.57 39H	.58 3AH	.59 3BH
.60 3CH	.61 3DH	.62 3EH	.63 3FH	.64 40H	.65 41H	.66 42H	.67 43H	.68 44H	.69 45H	.70 46H	.71 47H
.72 48H	.73 49H	.74 4AH	.75 4BH	.76 4CH	.77 4DH	.78 4EH	.79 4FH	.80 50H	.81 51H	.82 52H	.83 53H
.84 54H	.85 55H	.86 56H	.87 57H	.88 58H	.89 59H	.90 5AH	.91 5BH	.92 5CH	.93 5DH	.94 5EH	.95 5FH
.96 60H	.97 61H	.98 62H	.99 63H	.100 64H	.101 65H	.102 66H	.103 67H	.104 68H	.105 69H	.106 6AH	.107 6BH
.108 6CH	.109 6DH	.110 6EH	.111 6FH	.112 70H	.113 71H	.114 72H	.115 73H	.116 74H	.117 75H	.118 76H	.119 77H
.120 78H	.121 79H	.122 7AH	.123 7BH	.124 7CH	.125 7DH	.126 7EH	.127 7FH	.128 80H	.129 81H	.130 82H	.131 83H
.132 84H	.133 85H	.134 86H	.135 87H	.136 88H	.137 89H	.138 8AH	.139 8BH	.140 8CH	.141 8DH	.142 8EH	.143 8FH
.144 90H	.145 91H	.146 92H	.147 93H	.148 94H	.149 95H	.150 96H	.151 97H	.152 98H	.153 99H	.154 9AH	.155 9BH
.156 9CH	.157 9DH	.158 9EH	.159 9FH	.160 A0H	.161 A1H	.162 A2H	.163 A3H	.164 A4H	.165 A5H	.166 A6H	.167 A7H

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 2 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 0112228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke, Hans-Joachim Hill (Tel. 2870203); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelbild Stefan Seeboldt, Thomas Neumann

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

Redaktionsschluß: 13. Dezember 1988

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

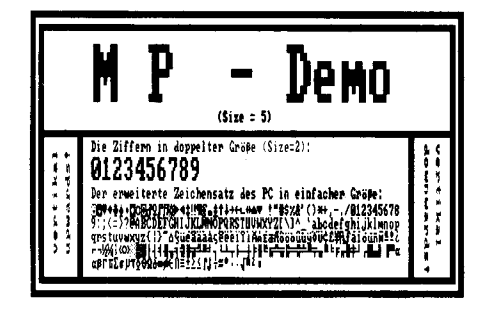
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandittit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 12000 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 88547 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Der Beitrag „Diskettenformate CP/M-kompatibler Betriebssysteme" auf der **Seite 35** beschäftigt sich mit dem physischen und logischen Aufbau von Disketten. Es werden Hilfestellungen bei Schwierigkeiten mit Disketten und Tips für den Umgang mit Ihnen gegeben.

Auf der **Seite 39 werden in dem Artikel „Anwendung des FDC U 8272 D" einige Hinweise zum zusätzlich von diesem Schaltkreis unterstützten Aufzeichnungsformat 128 Byte/Sektor MFM gegeben.** Weiterhin wird gezeigt, wie man verfahren kann, wenn man Disketten mit mehr als 77 Spuren benutzt; ein digitaler Datenseparator und eine Prekompensationsschaltung werden diskutiert.



Auf der **Seite 53** finden Sie den Beitrag „8 × 8 – ein Font für alle Fälle", der einen Font vorstellt, der gegenüber dem 4 × 6-Font bei höher auflösender Grafik eine bessere Konturschärfe bei gleicher Schriftgröße bietet. Der Font sowie die Bildschirmausgabe des DEMO-Programms zum 8 × 8-Pixelzeichensatz sind auf **der 2. und 3. Umschlagseite** dargestellt.

Vorschau

Für MP 3/1989 haben wir für Sie unter anderem Beiträge zu folgenden Themen vorbereitet:
- A 7100-Grafik
- Übergang von dBase II zu dBase III
- Basic-inline-Pokes.

Inhalt

DCP-Softwaremesse



Am 24. und 25. November 1988 zeigte der VEB Robotron-Projekt gemeinsam mit anderen Betrieben des Kombinates in Dresden, Am Fučikplatz (Halle A), die Leistungsfähigkeit seiner angebotenen Softwareprodukte. Diese erste Softwaremesse für die Rechnertypen EC 1834 und A 7150 zog über 6500 Interessenten an. Neben einer übersichtlichen Gestaltung der Schau standen stets kompetente Mitarbeiter an fast 50 Personalcomputern zur Verfügung, die Auskünfte zu ihren Produkten gaben.
In acht Themenkomplexen konnten die Anwender jeweils die für sie optimale Problemlösung auswählen und begutachten:
- Betriebssysteme und Programmiersprachen
- Textsysteme/Integrierte Systeme
- Datenbanksysteme/Informationsrecherchesysteme
- Mathematische Verfahren/Kalkulation/Geschäftsgrafik
- CAD-Software
- Datenkommunikation/Lokale Netze
- Technologie/Künstliche Intelligenz
- Anwendersysteme.

Die Softwarebetriebe des Kombinates Robotron demonstrierten in den einzelnen Komplexen wirklich attraktive und leistungsfähige Lösungen und ihre Einsatzfähigkeit auf der modernen in der DDR produzierten Hardware. Neben vielen bekannten und bereits weit verbreiteten Softwarepaketen waren Neuerungen von hohem Niveau und mit guter Bedienoberfläche für DCP zu sehen, wie das bilinguale Textsystem TEXT 40 K, ein Window-Editor, AIDOS/M1 und viele andere. Unter der Überschrift „Anwendersysteme" demonstrierten über 30 weitere Betriebe ihre Hardware- und Softwarelösungen unter Anwendung der Robotron-Systeme. Das Thema „Nachnutzung" wurde an jedem dieser Stände heiß diskutiert.
Materialien zur Ausbildung an und mit dem Personalcomputer boten das Robotron-Schulungszentrum Leipzig mit seiner Teachware sowie das Institut für Film, Bild und Ton Berlin mit Folien, Dias, Filmen, Videos und Lehrsoftware.
Über die Leistungsfähigkeit der angebotenen Produkte informierten sich auch der 1. Sekretär der Bezirksleitung Dresden der SED, Hans Modrow, sowie die in der DDR weilende Delegation des europäischen Parlaments. Eine nächste Schau dieser Art wird es im April 1990 auf der doppelten Fläche geben, auch das ist ein Ergebnis der erfolgreichen Messe.

M. Fischer

MP im Jahr 1989

Vielleicht sollte die Redaktion darüber nachdenken, in jedem Heft einen ausführlicheren Hardwareartikel zu veröffentlichen. – Eine stärkere Ausrichtung in Richtung Software wäre aus unserer Sicht wünschenswert. – ... doch wäre ich auch erfreut, öfter mal einen kompletten Schaltplan zur praktischen Nachnutzung zu finden. – Vielleicht ist es auch möglich, mehr Quelltexte in höheren Programmiersprachen abzudrucken. – Nicht einverstanden bin ich damit, daß immer wieder wertvoller Platz mit BASIC-Sprachübersichten ... verschwendet wird. – Die Attraktivität der MP leidet stark durch die vielen Kleincomputer-Artikel. – Besonders interessiere ich mich für den Einsatz der Kleincomputer und deren Hard- und Software. – ... glaube ich, daß der Kleincomputer in der MP zu stark vertreten ist, ... es gibt z. B. sehr wenig Material über die verschiedenen PC-Betriebssysteme, und gerade das könnte ich für die Bewältigung meiner Arbeitsaufgaben gut gebrauchen. – Wo bleiben in der MP eigentlich die Amateure? Soll das, was unter der Rubrik „MP-Computer-Club" erscheint, alles sein?

„Jedermann recht getan ..." werden vielleicht auch Sie nach der Lektüre dieser kleinen Auswahl von Leserstimmen denken, die uns im Laufe des vergangenen Jahres erreichten. Um mit dem letzten Zitat zu beginnen: Das, was in der Rubrik Computer-Club zum Thema Kleincomputer erscheint, muß vom Umfang her tatsächlich alles sein. Nicht, daß wir die Bedeutung unterschätzten, die der Computer heute im Freizeitbereich hat. Jedoch gilt es, die Aufgabe einer wissenschaftlich-technischen Fachzeitschrift zu erfüllen, die da lautet: Unterstützung eines berufsmäßig vorgebildeten Leserkreises in seiner praktischen Tätigkeit. Für die MP bedeutet dies: Unterstützung aller ingenieurtechnischen Kader, die auf dem Gebiet der Mikroelektronik, Computertechnik und Informatik tätig sind, mit anwendungsbezogenen Beiträgen zu ihrem Arbeitsgebiet. Es entspricht dabei sowohl der Erwartung unserer Leser – siehe oben – wie auch der uns gestellten Aufgabe, die Anwendung der Mikroelektronik als *Einheit* von Bauelementen, Baugruppen und Programmsystemen darzustellen.
Erwarten können Sie also auch weiterhin, daß die MP über die spezifischen Bauelemente der Computertechnik und deren Applikation, über Entwicklungen auf dem Computersektor – etwa zu gleichen Teilen Hardware und Software – sowie zukunftsorientierend zu Fragen der Informatik berichten wird. Dabei werden wir jene Beitragsangebote bevorzugen, in denen auch komplizierte Vorgänge anschaulich und für den Praktiker verständlich dargestellt sind.
Das betrifft sowohl Übersichtsbeiträge zu allgemein interessierenden Fragen der Informatik als auch Artikel im Stil von „Wie wird's gemacht?", zu deren Einsendung wir in MP 2/88 aufriefen. In dem insgesamt sehr großen Manuskriptangebot sind die Beiträge leider noch etwas rar, die den Weg zur Lösung eines Problems leicht verständlich darlegen oder die eine Entwicklung unmittelbar zur Nachnutzung offerieren.
Noch stärker als bisher werden wir bei Manuskriptangeboten zur Vorstellung von Hard- oder Softwarelösungen, die Übersichtscharakter haben, auf ein kurzgefaßtes Angebot für die Rubrik Börse dringen müssen.

Als kleine Auswahl aus dem breit gefächerten Angebot dieses Jahres wären abschließend zu nennen:

- Beiträge zu wichtigen Entwicklungen unserer Bauelementeindustrie anläßlich des 13. Mikroelektronik-Bauelementesymposiums
- Beiträge zu Rechnerarchitekturen, integrierten Softwarepaketen und zur Grafik am A 7100/7150
- in unserer Reihe KURS die Programmiersprache Forth und perspektivisch MS-DOS, Unix oder Modula-2.

Mit dieser Zielsetzung hoffen wir, die MP in Ihrem Sinne künftig noch interessanter und lehrreicher gestalten zu können.

Ihre Redaktion MP

20 Jahre Rank Xerox EEO

Im Jahre 1968 gründete die Rank Xerox Ltd. – mit Hauptverwaltung in London und zuständig für Europa, Afrika und Australien – ihre Spezialabteilung Eastern Export Operations (EEO). Anläßlich dieses Jahrestages gab das Rank Xerox Büro Berlin/DDR ein 12seitiges Material heraus, das mit dem DTP-System Ventura Publisher erstellt und auf dem Laserdrukker RX 4045 ausgegeben wurde.
In der DDR ist Rank Xerox seit mehr als zwanzig Jahren tätig und seit 1981 als Firma akkreditiert. In dieser Zeit hat sich eine gute Zusammenarbeit mit der Vertreterfirma F. C. Gerlach und dem VEB Robotron Vertrieb entwickelt, der den Service für Rank Xerox-Geräte übernommen hat.
Als traditioneller Hersteller von Kopiertechnik ist Rank Xerox bekannt, jedoch entstanden in den Forschungs- und Entwicklungslabors der Firma auch das bekannte lokale Netz Ethernet, der Laserdruck und die Idee der Maus als Eingabemedium für Mikrocomputer. Mit dem frühzeitigen Erwerb der Rechte an dem DTP-Programm Ventura Publisher ebnete auch Rank Xerox dem „Publizieren am Schreibtisch" den Weg.
Heute ist das Unternehmen darum bemüht, sich auf dem Gebiet der Datentechnik insgesamt zu profilieren. Dementsprechend befinden sich neben Kopier- und Vervielfältigungssystemen auch Drucker, elektronische Schreibmaschinen, IBM-kompatible PCs, Grafikbildschirme, lokale Netze sowie Software wie Ventura Publisher (seit kurzem in der neuen Version 2.0, siehe MP 1/1989, S. 30) im Vertriebsprogramm.

MP

China intensiviert Computerproduktion

Die VR China plant in den nächsten Jahren ein Steigerung der Jahresproduktion von Computern um nahezu das Dreifache. Bis 1990 sollen jährlich Rechner im Wert von sechs Milliarden Yuan hergestellt werden. Wie Hsinhua am Dienstag meldete, verzeichnete die Computerindustrie von 1981–1987 durchschnittlich einen Zuwachs von 30 Prozent pro Jahr. 1987 wurden Computer mit einem Wert von 2,1 Milliarden Yuan hergestellt. Obwohl China gegenwärtig mittlere und Großcomputer – darunter solche zur Verarbeitung chinesischer Schriftzeichen – in mehreren Varianten herstellen könne, sei die internationale Wettbewerbsfähigkeit weiterhin gering. Der Ausbau des Industriezweiges soll auch zu einer Qualitätssteigerung in diesem Bereich beitragen und den Absatz auf Überseemärkten fördern.

ADN

Diskettenformate CP/M-kompatibler Betriebssysteme

Dr. Wilfried Dames
Akademie der Wissenschaften der DDR, Institut für Informatik und Rechentechnik

Auf dem internationalen Markt gibt es weit über hundert Diskettenformate. Disketten können single- oder double-density und single- oder double-sided benutzt werden, die Spuranzahl kann 35, 40, 70 oder 80 betragen, in einem Sektor können 128, 256, 512 oder 1024 Bytes stehen und die Sektor- und Spurreihenfolge kann variieren. Die Blockgröße reicht von 1 KByte über 2 KByte, 4 KByte bis zu 8 KByte bei 32, 64, 96, 128, 192 oder 256 Verzeichniseinträgen und 0 bis 5 Systemspuren. Mit dem folgenden Beitrag soll versucht werden, dieses Chaos etwas zu entwirren und Einblick in die Arbeitsweise des Betriebssystems zu geben.

Disketten sind mittlerweile das gängige Speichermedium für Personal- bzw. Arbeitsplatzcomputer und finden auch im Homecomputer-Bereich zunehmende Verbreitung. Sie haben eine handliche Größe und arbeiten bei vernünftigem Umgang recht zuverlässig. Doch spätestens wenn man von seiner Diskette außer *Bad Sector* keine weitere Information mehr bekommt oder das Verzeichnis einer eigentlich vollen Diskette *NO FILE* lautet, muß man sich wohl oder übel mit dem physischen und logischen Aufbau von Disketten beschäftigen. Im folgenden sollen dabei einige Hilfestellungen und Hinweise gegeben werden.

Technisch bedingte Diskettenparameter

Obwohl dieser Abschnitt fast nichts mit CP/M zu tun hat, basiert der größte Teil der Formatvielfalt auf den technischen Möglichkeiten von Diskettenlaufwerken und der dazugehörigen Ansteuerelektronik (englisch: controller).

Informationen werden auf einer Diskette in konzentrischen Spuren (englisch: tracks) gespeichert. Auf Grund des magnetischen Aufzeichnungsverfahrens sind sie nicht sichtbar. Außerdem ist ihre Lage und Größe völlig frei innerhalb der Magnetschicht wählbar. Moderne Disketten (und die dazugehörigen Laufwerke) haben eine so hohe Qualität, daß man 80 und mehr Spuren auf einer Breite von zirka 3 cm unterbringen kann. Die Informationen werden analog einem Kassettengerät durch einen kombinierten Schreib-/Lese-/Löschkopf aufgezeichnet und auch gelesen. Dieser Kopf ist auf einem Schlitten von außen nach innen und zurück verschieblich und legt durch seine Position die Lage der Spur fest. Extrem geringe Toleranzen beim direkten Positionieren auf eine Spur und die Spaltgröße des Kopfes bestimmen wesentlich die mögliche Anzahl der Spuren und damit die Aufzeichnungskapazität.
Eine Diskette hat für gewöhnlich 40 oder 80 physische Spuren. Es müssen nicht alle 40 oder 80 möglichen Spuren benutzt werden, auch 35 bzw. 70 Spuren sind üblich. In Zusammenhang mit Kopierschutzmethoden werden oft auch unter Ausnutzung technischer Reserven mehr als 40 oder 80 Spuren verwendet.

Entsprechend dem Durchmesser der Diskette unterscheidet man 8″ und 5,25″ Disketten, aber auch kleinere Disketten mit stabilerer Außenhülle existieren als Speichermedium. Auf Grund der großen Verbreitung in der DDR werden im folgenden vorrangig 5,25″ Disketten betrachtet, für andere Typen gelten die Aussagen sinngemäß.
Eine Diskette kann einseitig (englisch: Single Sided – SS) oder zweiseitig (englisch: Double Sided – DS) benutzt werden. Letzteres ist sowohl eine Frage der Diskette selbst (Magnetschicht auch auf der anderen Seite) als auch der Laufwerke (Kopf auch auf der anderen Seite). Sollte man einmal in die Verlegenheit kommen, einen Kratzer oder Fleck zu suchen, der zu *Bad Sector* führt, so ist die Vorderseite (oder die einzige Seite) einer Diskette im allgemeinen deren Rückseite, also die Seite ohne Aufkleber! Weiterhin sei an dieser Stelle darauf hingewiesen, daß eine zweimal einseitige Diskette (Umdrehen der Diskette) auf der zweiten Seite in einer anderen Richtung beschrieben wird als eine zweiseitige Diskette (ohne Umdrehen), man benötigt also für letztere immer ein doppelseitiges Laufwerk!
Die Informationen werden auf einer Spur durch magnetische Flußwechsel aufgezeichnet. Von den vielen Möglichkeiten haben zwei Standards die größte Bedeutung: die einfache Dichte (englisch: Single Density – SD; vorrangig bei 8″-Disketten) und die doppelte Dichte (englisch: Double Density – DD; vorrangig bei 5,25″-Disketten). Andere Aufzeichnungsverfahren nutzen z. B. die Tatsache aus, daß die äußeren Spuren länger, das heißt mehr Flußwechsel pro Millimeter möglich sind. Sie werden jedoch im Personal- bzw. Arbeitsplatzcomputer-Bereich in der DDR nicht verwendet. Aus Kompatibilitätsgründen unterstützen derartige Rechner i. a. auch eines der oben angegebenen Aufzeichnungsverfahren.

Aus den technischen Daten von Laufwerken soll nun einmal die physische Spurkapazität berechnet werden. Die Diskette dreht sich mit der genormten Geschwindigkeit von 300 Umdrehungen pro Minute oder 300/60 = 5 Umdrehungen pro Sekunde oder 1000/5 = 200 Millisekunden pro Umdrehung. Die Zeit für die Aufzeichnung eines Bits beträgt 4 Mikrosekunden, das heißt, die Bitfrequenz 250 kHz (dies ist übrigens die mit Disketten erreichbare Datenrate von 250 kBit/s). Da jeweils 8 Bits ein Byte bilden, ergibt sich die physische Spurkapazität zu 200 ∗ 250/8 = 6250 Bytes. Auf eine zweiseitige 80-Spur-Diskette passen damit physisch 2 ∗ 80 ∗ 6250 = 1.000.000 Bytes.

Leider ist diese physische Spurkapazität nicht vollständig für Daten nutzbar. Um den Direktzugriff zu realisieren, um eine gewisse Datensicherheit zu gewährleisten und auf Grund von Hard- und Softwarereaktionszeiten sind eine Reihe von Dienstinformationen und sogenannten Lücken (englisch: gap) auf jeder Spur notwendig. Von Bedeutung ist dabei, daß die Spur in Sektoren unterteilt wird. Jeder Sektorbeginn wird hardwaremäßig durch eine spezielle Aufzeichnung, sogenannte Marken (englisch: mark) gekennzeichnet und kann ohne Softwareunterstützung aufgefunden werden. Unmittelbar danach beginnt der Sektorkopf (auch Kennzeichnungsfeld genannt), in dem Spurnummer, Kopfnummer, Sektornummer und Sektorlänge stehen. Nach einer kurzen Lücke (um Zeit zur Auswertung dieser Informationen zu lassen) folgen die eigentlichen Daten, die jeweils eine Länge von 128, 256, 512 oder 1024 Byte haben. Sowohl Sektorkopf als auch Sektordaten sind jeweils mit 2 Kontrollbytes (englisch: Cyclic Redundance Check – CRC) gegen Aufzeichnungsfehler abgesichert. Nach den beiden CRC-Bytes für die

Kennzeichnung der Disketten

Für die fehlerfreie Arbeit mit Disketten ist die Kenntnis ihrer Kennzeichnungen erforderlich. Die folgende Tafel führt die möglichen Abkürzungen und ihre Bedeutung auf.

Abkürzung	Bedeutung
HD	High Density – hohe Speicherdichte
KB	1 Kilobyte = 1024 Byte
MB	1 Megabyte = 1024 KByte
Abkürzung	**Bedeutung bei 3,5″-Disketten**
1	einseitig geprüfte Diskette
2	zweiseitig geprüfte Diskette
DD	doppelte Aufzeichnungsdichte
HD	High Density-Aufzeichnungsdichte
Abkürzung	**Bedeutung bei 5,25″-Disketten**
1S	einseitig geprüfte Diskette
2S	zweiseitig geprüfte Diskette
2D	doppelte Aufzeichnungsdichte
96tpi	Schreibdichte 96/100 Spuren pro Zoll
HD	High Density-Aufzeichnungsdichte

Bei Kenntnis dieser Angaben kommt man natürlich auch leicht darauf, Disketten in gewissem Sinne zweckentfremdet einzusetzen. Man kann zum Beispiel bei einem einseitigen Laufwerk (Laufwerk mit nur einem Magnetkopf) die Disketten auch zweiseitig verwenden. Dazu muß natürlich symmetrisch zur Schreibschutzkerbe eine zweite eingeschnitten werden. Eventuell ist auch symmetrisch zum ersten ein zweites Indexloch nötig. Das hat allerdings zur Folge, daß der Andruckfilz (gegenüber dem Magnetkopf) ständig auf der nicht benutzten Folienseite reibt und diese wiederum nach Umdrehen der Diskette den Magnetkopf bearbeitet. Das senkt natürlich die Lebensdauer von Diskette und Magnetkopf.

Andersherum ist es aber möglich, einseitig geprüfte (billigere) Disketten in einem zweiseitigen Laufwerk zu benutzen. Die nicht geprüfte zweite Seite kann völlig in Ordnung sein oder kann eventuell mit weniger Speicherkapazität als die erste Seite verwendet werden.

MP

Daten befindet sich eine sehr wichtige Lücke bis zum Beginn des nächsten Sektors. Durch sie wird beim Schreiben von Daten im Direktzugriff (also mitten auf der Spur) verhindert, daß trotz Umdrehungsschwankungen des Laufwerks oder beim Wechsel zu einem anderen Rechner diese Daten den nächsten Sektor überschreiten. Übliche zulässige Gleichlaufschwankungen sind zirka 2–3%, moderne Laufwerke haben jedoch weitaus geringere Toleranzen (etwa 0,5%). Durch diese Schwankungen entsteht am Ende jedes Sektors übrigens ein ziemliches Durcheinander von Bitresten der vorhergehenden Aufzeichnungen, die nicht mehr im genormten Flußwechsel-Abstand liegen. Auch deshalb ist es notwendig, die Hardware wieder durch die nächste Sektorbeginn-Marke auf den Sektoranfang zu synchronisieren.
Die auf eine Diskette passende Informationsmenge kann man nun dadurch beeinflussen, daß man
– längere Sektoren schreibt, um weniger Lücken zu haben
oder
– die Lücken zwischen den Sektoren auf das notwendige Mindestmaß beschränkt.
Nimmt man an, daß alle Sektoren einer Spur gleich lang sind, so teilt sich die Kapazität einer Spur nach folgender vereinfachter Formel auf:

kapazität ≧ spurkopflänge + sektorzahl * (sektorkopflänge + datenlänge + lücke)

Die Längen werden jeweils in Byte angegeben; bei den Kopflängen werden auch alle konstanten Informationen zwischen Kopf und Daten und die CRC-Bytes nach den Daten erfaßt. Der Spurkopf kennzeichnet den Anfang einer Spur, Aufbau und Länge sind durch das Aufzeichnungsverfahren vorgeschrieben.
Aus obiger Formel läßt sich leicht die Formel für die maximale Lücke zwischen Daten und nächstem Sektor ableiten:

lücke ≦ (kapazität – spurkopflänge)/ sektorzahl – sektorkopflänge – datenlänge

Es wird vereinfachend angenommen, daß der Formatiervorgang auf einem Laufwerk mit 0% Schwankung erfolgte, so sind Spur- und Sektorköpfe fest auf der Spur und nur die Datenlänge schwankt beim Schreiben auf der Diskette. Sie wird daher für 2% Gleichlaufschwankung mit dem Faktor 1,02 multipliziert.
In der Tafel 1 werden die maximalen Lücken für einige gängige Formate berechnet. Mit (*) ist dabei dasjenige Format gekennzeichnet, daß die größte Datenkapazität pro Spur (= sektorzahl * datenlänge) liefert. Daraus läßt sich die Datenkapazität einer Diskette durch Multiplikation mit der Seitenzahl und der Spuranzahl errechnen. Sie beträgt z. B. günstigenfalls bei 5,25", DD, DS, 80 Spuren 5 KByte * 2 * 80 = 800 KByte.
Datenlängen von 128 Byte sind bei 5,25", DD unüblich und stehen daher nicht in Tafel 1. (Wie der FDC U 8272 dieses Format unterstützt, wird im nachfolgenden Beitrag erläutert. Die Red.)
Das Zusammenspiel von Sektorzahl, Datenlänge und Lückenlänge ist bei den meisten Systemen durch das jeweilige Format-Programm festgelegt und kann vom Anwender nur bei wenigen Systemen beeinflußt werden. Gerade hierin stecken jedoch, wie Tafel 1 zeigt, erhebliche Reserven für die Datenkapazität einer Diskette.

Hinweise zum Umgang mit Disketten

Damit Sie immer Freude an der Arbeit mit Disketten haben, wollen wir Ihnen, falls Sie sich nicht schon zu den Profis zählen, einige Hinweise geben.

- Generell sollte man von den gerade erstellten Programmen oder Daten eine Sicherheitskopie anfertigen, z. B. vor jedem Feierabend.

- Sollten Sie ein Laufwerk mit hoher Spurdichte verwenden, dann ist von Billigdisketten abzuraten. Die Magnetschicht der Disketten ist flexibel, und die Laufwerksjustage läßt durch den dauernden Diskettenwechsel nach. Bei billigen Disketten erhöht sich die Gefahr, daß der Magnetkopf die Spur verfehlt.

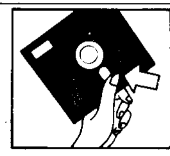
- Das Antriebsloch der 5¼"-Disketten besitzt einen Verstärkungsring (Hard Hole). Er dient der Verbesserung des Antriebs und der Zentrierung im Laufwerk. Disketten, die diesen Ring besitzen, können in den 5¼"-Laufwerken benutzt werden, wenn die Betriebsanleitung für diese Laufwerke das nicht ausdrücklich verbietet.

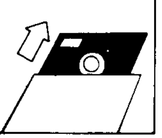
- Disketten, die nicht nur ein Loch in der Magnetfolie, sondern viele Löcher in regelmäßigen Abständen besitzen, sind hardsektorisierte Disketten. Sie können nur in speziellen (älteren) Laufwerken verwendet werden.

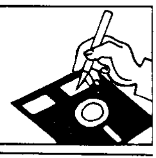
- Der Hinweis: *sektors soft* auf der Diskette kennzeichnet nur, daß diese Diskette noch nicht formatiert (softsektoriert) ist.

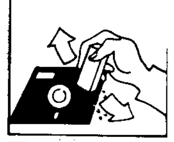
Weitere Tips finden Sie in der Bildserie.

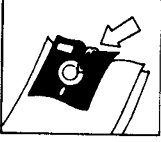
MP

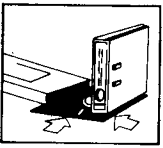

*Die Diskette **nicht** an ihrer magnetischen Oberfläche berühren.*

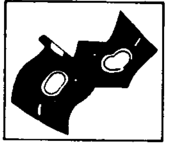

*Die Diskette **nicht** unnötig aus der Schutzhülle nehmen.*

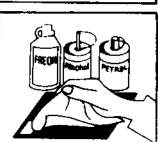

***Keine** spitzen Bleistifte oder Kugelschreiber für das Beschriften verwenden.*


***Keinen** Radiergummi verwenden.*

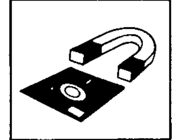

***Keine** Büroklammern, Heftklammern oder Gummis an der Diskette befestigen.*

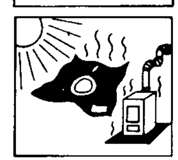

***Keine** schweren Gegenstände auf die Diskette stellen.*

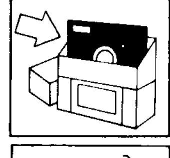

*Disketten **nicht** biegen oder falten.*

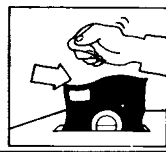


*Die Diskette **nicht** mit Lösungsmitteln in Berührung bringen.*


*Magnetische Gegenstände bzw. Magnetfelder von der Diskette **fernhalten**.*


*Die Disketten **nicht** Temperaturen über 53 °C oder unter –40 °C aussetzen und **nicht** über 50 °C bzw. unter 10 °C einsetzen.*


*Disketten **aufrecht** lagern.*


*Beim Einführen ins Laufwerk **keine** Gewalt anwenden.*

Zugriff auf die Daten

Der eigentliche Datenzugriff wird vom BIOS des CP/M realisiert. In dessen hardwareabhängigen Steuerprogrammen sind alle Routinen zum Auswählen des physischen Laufwerks, zur Spurpositionierung und zum Lesen und Schreiben konzentriert. Im allgemeinen arbeitet das BIOS mit einem oder mehreren Diskettenpuffern, da die physische Sektorgröße von beispielsweise 1024 Byte (1 K) von CP/M 2.2 nur in Abschnitten von 128 Byte (logischer CP/M-Sektor) verarbeitet werden können.
Die Diskettenlaufwerke und deren Ansteuer-

Tafel 1 *Lückenlänge für unterschiedliche Formate, alle Längenangaben in Byte*

Typ	spur-kapazität	spur-kopf-länge	sektor-kopf-länge	daten-länge	sektor-zahl	Lücke bei ±0%	Schwankung ±2%
5,25" DD	6250	146	62	256	18	21	15
					17	41	35
					16	63	58
					15	88	83
				512	10	36	26 (*)
					9	104	93
					8	189	178
				1024	5	134	114 (*)
8" SD	5208	73	33	128	26	36	33
				256	17	13	7
					16	31	26
					15	53	48
				512	9	25	15 (*)
					8	96	86
				1024	4	226	206
8" DD	10416	146	62	256	31	13	8
					30	24	19
					29	36	31
					26	77	71
				512	17	30	19
					16	67	57
					15	110	100
				1024	9	55	34 (*)
					8	197	177

technik bieten zur Erhöhung der Zugriffsgeschwindigkeit und Unterstützung der Datenverwaltung auf der Diskette zwei weitere Variationsmöglichkeiten, die auch tatsächlich ausgenutzt werden:

– Die Sektoren können innerhalb der Spur in beliebiger Reihenfolge stehen (englisch: Skew-Faktor oder Interleave)

und

– bei zweiseitigen Disketten kann der Seitenwechsel beim Datenzugriff zu beliebigen Zeitpunkten erfolgen.

Die Sektorreihenfolge bestimmt wesentlich die Geschwindigkeit beim kontinuierlichen Lesen von Daten (z. B. Laden eines Programms oder Textes). Dabei werden die Sektoren so angeordnet, daß die nächsten benötigten Daten gerade dann bereitstehen, wenn der Rechner die vorherigen Daten verarbeitet und den nächsten Leseauftrag vorbereitet hat. Diese Zeit hängt nun von sehr unterschiedlichen Faktoren ab, z. B.

- Prozessorgeschwindigkeit
- Art des Datentransfers (parallel zum Prozessor über DMA oder unter Prozessorsteuerung)
- Leistungsfähigkeit der Floppy-Ansteuerung (z. B. automatische CRC-Berechnung und -Kontrolle ohne Mitwirkung des Prozessors)
- Puffergröße im BIOS (wieviele Sektoren können ohne logische Auswertung unmittelbar hintereinander gelesen werden).

Da jedes Rechnersystem eigene Leistungsparameter hat, gibt es keine optimale Sektorreihenfolge. Auch ist der Abstand zwischen den Sektornummern oft nicht konstant, da bei Diskettenformaten mit einer Sektorlänge, die kleiner als die Länge des BIOS-Puffers ist, mehrere Sektoren hintereinander direkt vom Floppy-Controller mit automatischer Sektornummernerhöhung in den BIOS-Puffer gelesen werden und dann eine Pause für die Verarbeitung der Daten im Puffer notwendig ist.

Einige Beispiele sollen das belegen (siehe Tafel 2), wobei man sich jeweils die Sektoren im Kreis angeordnet vorstellen muß. Sie wurden durch Messungen an Rechnersystemen mit den angegebenen Leistungsparametern empirisch ermittelt.

Leider gibt es nun prinzipiell zwei Methoden, um diese Sektorreihenfolge zu erreichen:

– monotone Anordnung der Sektoren auf der Spur und Benutzung einer Übersetzungstabelle für die Sektornummern beim Zugriff

oder

– physische Anordnung der Sektoren auf der Spur in der gewünschten (optimalen) Reihenfolge.

Die erste Methode hat im Gegensatz zur zweiten den Nachteil, daß man diese Tabelle kennen muß, um die Sektoren in der richtigen logischen Reihenfolge verarbeiten zu können. Sie war besonders bei 8"-Laufwerken mit dem Format 26 * 128 üblich, wo ein konstanter Sektorversatz von 6 angenommen wurde (also 1, 7, 13, ...) und findet ihren Niederschlag in der CP/M-BIOS-Funktion *sectortranslate*. Kennt man diese Tabelle nicht, so hilft nur eine genaue Analyse der Daten auf der Diskette, z. B. mit dem READ-Kommando von POWER oder DIENST. Dabei sollte man sich möglichst eine Diskette mit Texten vornehmen, da hierbei die Textfortsetzung innerhalb der Spur besonders leicht gefunden werden kann. Ist das Original-Rechnersystem zugänglich, so ist es günstiger, dort ein „systematisches" File zu erzeugen, wo in jedem logischen 128er CP/M-Sektor eine laufende Nummer abgelegt wird, z. B. mit Hilfe von POWER:

```
FILL 4000 4FFF 0
DS 4000
1
DS 4080
2
...
DS 4F80
20
SAVE TEST 4000 32 oder WRITE spur 1 4000 32
```

Um nun die so erkannte Sektorreihenfolge auch tatsächlich zu unterstützen, benötigt man ein CP/M-System, in dem die Floppy-Parameter geändert werden können (per Dienstprogramm oder durch Patchen der Steuertabellen).

Mit Hilfe der Sektornummern-Übersetzung kann man jedoch auch die physischen Sektoren auf der Spur ganz willkürlich numerieren, so z. B. nicht wie üblich ab 1, sondern ab 0 (195 KByte KAYPRO, ab 41H (171 KByte Schneider, System) oder C1H (180 KByte Schneider, Daten) oder auf der Spurrückseite anders als auf der Spurvorderseite. Die Absicht dieser Formate ist sicher nicht allein eine Optimierung des Zugriffs. Neben dem beliebten Begriff des *Hausformats* stecken zum Teil auch Methoden zur automatischen Formaterkennung dahinter, da CP/M diesbezüglich auf Grund seiner Historie (in den 70er Jahren gab es nur ein Format, nämlich 8" SD, SS mit 26 Sektoren zu je 128) keine Unterstützung bietet.

Im folgenden wird die Art und Weise des Seitenwechsels bei zweiseitigen Disketten betrachtet. Auch hier ist alles anzutreffen, was erlaubt und einigermaßen sinnvoll ist:

- Spur 0 Vorderseite, Spur 0 Rückseite, Spur 1 Vorderseite, ...
  Diese Variante ist die häufigste, da sie bei dicht liegenden Daten wenig Positioniervorgänge erfordert.
- Spur 0 Vorderseite, ..., Spur 79 Vorderseite,
  Spur 79 Rückseite, ..., Spur 0 Rückseite.
  Hierdurch lassen sich zweiseitige Disketten logisch wie doppelt so große einseitige behandeln (CP/M 86).
- Spur 0 Vorderseite, ..., Spur 79 Vorderseite,
  Spur 0 Rückseite, ..., Spur 79 Rückseite.
  Diese Variante ist durch den Sprung von ganz innen nach außen beim Seitenwechsel relativ ungünstig.

Tafel 2 *Sektorreihenfolgen bei verschiedenen Spurformaten und Rechnersystemen*

Spurformat 5 * 1024

Rechnersystem:
2,5-MHz-U 880, Transfer und CRC durch Prozessor, 1 KByte Puffer
Sektorreihenfolge:
1, 5, 4, 3, 2 (3,5 Umdrehungen für das Lesen einer Spur)

Rechnersystem:
4,0-MHz-U 880, Transfer durch Prozessor, CRC durch Controller, 1 KByte Puffer
Sektorreihenfolge:
1, 4, 2, 5, 3 (2 Umdrehungen für das Lesen einer Spur)

Spurformat 16 * 256

Rechnersystem:
2,5-MHz-U 880, Transfer und CRC durch Prozessor, 1 KByte Puffer
Sektorreihenfolge:
1, 2, 3, 4, 13, 14, 15, 16, 9, 10, 11, 12, 5, 6, 7, 8 (3,5 Umdrehungen für das Lesen einer Spur)

Rechnersystem:
4,0-MHz-U 880, Transfer durch Prozessor, CRC durch Controller, 0,25 KByte Puffer:
Sektorreihenfolge:
1, 9, 2, 10, 3, 11, 4, 12, 5, 13, 6, 14, 7, 15, 8, 16 (2 Umdrehungen für das Lesen einer Spur)

Darüber hinaus gibt es auch so abartige Methoden, daß wie bei den Sektornummern eine Spur-Übersetzungstabelle notwendig ist (z. B. Vorderseite 17...34, 16...0, Rückseite 17...34, 16...0 bei *Kontron* – oder Vorderseite 0,1, Rückseite 0,1, Vorderseite 2, Rückseite 2, ... bei *Alphatronic P3/P4*).

Welche Variante vorliegt, sieht man einer Diskette im allgemeinen von außen nicht an, hier helfen nur viele, viele Versuche ... – möglichst mit einer systematischen Datei, wie sie auch für die Sektorreihenfolge empfohlen wurde.

Logische Diskettenverwaltung unter CP/M

Nachdem in den vorherigen Abschnitten die wichtigsten technisch möglichen Varianten erläutert wurden, soll nun die Art und Weise der logischen Datenverwaltung unter CP/M 2.2 betrachtet werden.

CP/M 2.2 kennt intern nur die (logische) Sektorlänge von 128 Byte und verbirgt gegenüber dem Anwendungsprogramm völlig die physische Speicherung auf der Diskette. Wie Daten und Programme auf der Spur, von Spur zu Spur oder von Diskettenseite zu Diskettenseite abgelegt und wieder aufgefunden werden, steuert das BDOS von CP/M. In dieser klaren Trennung der physischen und logischen Rechnersystemeigenschaften liegt die Verbreitung und das große Softwareangebot von CP/M 2.2 begründet. Es ermöglicht, völlig unterschiedliche Datenträger wie 8"- und 5,25"-Diskettenlaufwerke, RAM-Floppys oder auch Festplatten mit verschiedenen Eigenschaften und Speicherkapazitäten gleichartig zu behandeln (trotz des zuvor angeführten Variantenreichtums).

Erreicht wird dies durch die im BIOS (das heißt im hardwareabhängigen CP/M-Teil) liegenden Disk-Parameter-Blöcke (DPB), deren Struktur feststeht, die jedoch in fast allen CP/M-Systemen um spezifische Informationen über das Diskettenformat und Laufwerkseigenschaften verlängert werden. Das BDOS benutzt nur den vorderen standardisierten Teil von 15 Byte. Eine intelligente Aufbereitung der Informationen im DPB bieten das CP/M-Hilfsprogramm STAT oder das Kommando DISK von POWER. Alle Angaben über das Diskettenformat entnimmt das BDOS ausschließlich dieser Tabelle, das heißt, dem BIOS!
Die Tafel 3 gibt eine kurze Übersicht über die DPB-Angaben.

Tafel 3 Disk Parameter Block

Offset	Bytes	Bedeutung
+ 0	2	*Anzahl der logischen 128er Sektoren pro Spur*
+ 2	1	*Anzahl der Bits, um die eine relative logische Sektornummer (Diskettenanfang = 0) nach rechts geschoben werden muß, um die Blocknummer zu berechnen*
+ 3	1	*Maske für die durch BSH verlorengegangenen Bits*
+ 4	1	*Anzahl der 16-KByte-Eintragungen (extents) pro Verzeichnis-Eintrag*
+ 5	2	*Anzahl der Blöcke – 1*
+ 7	2	*Anzahl der Verzeichniseinträge – 1*
+ 9	1 + 1	*pro belegtem Verzeichnisblock ein Bit gesetzt*
+11	2	*Zahl der logischen Verzeichnissektoren, die beim Diskettenwechsel überprüft werden sollen (z. B. 0 bei nicht wechselbarem Medium)*
+13	2	*Zahl der reservierten Spuren zu Beginn der Diskette*

Die Diskettenverwaltung erledigt das BDOS auf der Basis sogenannter Blöcke (bei POWER und DIENST *group* genannt), deren Größe in der Regel 1 KByte, 2 KByte oder auch 4 KByte beträgt und die (zum Leidwesen der BIOS-Systemprogrammierer) über Spur- und Seitengrenzen reichen können. Die gesamte für Daten nutzbare Diskettenkapazität, die wiederum aus dem DPB ermittelt wird, teilt das BDOS gleichmäßig in Blöcke auf. Zur Verwaltung des freien Speichers auf der Diskette dient ein sogenannter Disk-Allocation-Vector, der genau soviel Bits groß ist, wie Blöcke zu verwalten sind (0 = frei). Dieser Vektor liegt zur Beschleunigung des Zugriffs vollständig im Hauptspeicher, den Platz dafür muß das BIOS bereitstellen. Beim erstmaligen Zugriff zu einer Diskette (login) ermittelt das BDOS die Belegung der Tabelle aus dem Verzeichnis (englisch: directory), indem dort alle Blöcke aller existierenden Files als belegt gekennzeichnet werden. Dieser Vorgang wird insbesondere nach dem (lästigen) ^C bei Diskettenwechsel ausgelöst.
Da Disketten ohne Kenntnis des BDOS aus dem Laufwerk genommen werden können und damit eventuell bei der nächsten Diskette auf gar nicht freie Stellen geschrieben wird, überprüft das BDOS beim Eröffnen einer Datei jedesmal alle Verzeichniseintragungen durch eine Kontrollsumme und vergleicht sie mit den im Hauptspeicher beim Anlegen der Allocation-Tabelle hinterlegten Angaben über die Diskette. Stimmt dies nicht überein, so erfolgt die (berüchtigte) Meldung
BDOS ERR on X: Read Only,
denn gelesen werden kann auch von der neuen Diskette.

Wird ein freier Platz auf der Diskette gebraucht, so wird von vorn der erste freie BDOS-Block gesucht (Methode *first fit*, MS-DOS verwendet die günstigere Methode *next fit*). Auf leeren Disketten steht daher alles schön ordentlich hintereinander, auf einer Arbeitsdiskette dagegen ist nicht einmal gesagt, daß ein File von außen nach innen irgendwo liegt, es kann auch während der Fileerzeugung durch inzwischen gestrichene Hilfsfiles wieder weiter außen fortgesetzt werden! Dieser Hinweis sei all denen gegeben, die mal durch ein zerstörtes Verzeichnis ein 400-Teile Puzzle einer 800-KByte-Diskette zusammensetzen müssen. Dabei sollte man im übrigen mit Hilfe des READGR-Kommandos von POWER oder DIENST die Diskette analysieren, da ja die Filefortsetzung nur an Blockgrenzen losgehen kann und man daher nicht bei jedem beliebigen Sektor suchen muß!

Etwas unübersichtlich ist die Verzeichnisverwaltung von CP/M 2.2. Das Verzeichnis liegt immer in den ersten BDOS-Blöcken der Diskette (also zugriffsgünstig auf den äußeren Spuren und nicht in der Mitte). Jeder Verzeichniseintrag hat die feste Länge von 32 Byte, das heißt, es passen 4 Einträge in einen logischen Sektor, (siehe Tafel 4).

Tafel 4 CP/M-DIRectory-Aufbau

Offset	Bytes	Bedeutung
+0	1	*Nutzernummer 0 ... 1 FH oder E5H für frei*
+1	8	*Dateiname in ASCII nach rechts mit Leerzeichen aufgefüllt*
+9	3	*Dateityp in ASCII nach rechts mit Leerzeichen aufgefüllt*
+12	1	*Extension 0 ... 1 FH, Bit 5 – 7 = 0*
+13	1	*= 0 (Inhalt wird vom BDOS ignoriert)*
+14	1	*Extensionüberlauf bei Filelänge ≥ 512 KByte*
+15	1	*Anzahl der 128er Sätze im letzten Extent dieses Verzeichniseintrags*

Die ersten 16 Byte des Eintrags enthalten im wesentlichen den Filenamen, die nächsten 16 Byte die Liste der Blocknummern, auf denen das File gespeichert ist. Zwei Probleme hatten die CP/M-Entwickler dabei zu lösen:
- bei einer maximalen Anzahl von 256 Blöcken passen 16 Blöcke in einen Verzeichniseintrag, sonst aber nur 8 mit einer anderen Struktur von 2 Byte pro Block

und
- ein File kann länger sein, als Blöcke in einen Verzeichniseintrag passen.

Die Lösung des zweiten Problems waren sogenannte Extents, das heißt, es wurde einfach ein weiterer Verzeichniseintrag für das gleiche File angelegt, jedoch mit einer weitergezählten Extent-Nummer. Um wegen des ersten Problems die Software im BDOS nicht zu sehr zu belasten, ist ein Extent nicht identisch mit einem Verzeichniseintrag (!!), sondern immer 16 KByte groß, unabhängig davon, wieviele 16-KByte-Abschnitte in einem Verzeichniseintrag gespeichert werden können. Die Extent-Maske im DPB (offset +4) gibt nun an, wieviele Extents in einen Eintrag passen. Beim Schreiben eines Files gibt das BDOS übrigens jeweils nach 16 KByte den bis dahin angesammelten Verzeichniseintrag aus, auch wenn das Anwenderprogramm noch keinen Close-Befehl gegeben hat. Daher haben gerade geschriebene Files nach Systemabstürzen immer eine durch 16 K teilbare Länge, und man findet bis auf die letzten 16 KByte alle Daten; die fehlenden Datensätze liegen bis auf den Inhalt des letzten, noch nicht geschriebenen BIOS-Puffers, in den nächsten freien Blöcken der Diskette.
Woran erkennt man aber nun ein Fileende, ohne jedesmal das gesamte Verzeichnis durchsuchen zu müssen? Dazu dient das Verzeichnisbyte mit offset 15. Enthält es 80H, so ist der letzte Extent voll ($128 * 128 = 2^{14} =$ 16 K), und es gibt eine Fortsetzung des Files. Ist der Wert aber kleiner, so ist das File zu Ende.

Da CP/M keine hierarchischen Filestrukturen kennt, steht zur logischen Aufteilung „großer" Disketten die sogenannte Nutzernummer (user) zur Verfügung (eigentlich für Mehrplatzsysteme unter MP/M gedacht, daher der user-Begriff). Diese steht vor dem Dateinamen und kann 0 ... 15 (intern im BDOS übrigens 0 ... 31) betragen. Alle Angaben über 1 FH können keine Nutzernummer, also auch kein gültiger Verzeichniseintrag sein. Da beim Formatieren der Diskette E5H als Musterbyte vorgeschrieben ist, wird E5H auch gerade als Kennzeichen für einen freien Verzeichniseintrag benutzt, das heißt, eine frisch formatierte Diskette voller E5H-Bytes ist ohne weitere Aktionen leer. Wird ein File gelöscht, so wird die Nutzernummer einfach durch E5H überschrieben, Filename und sämtliche BDOS-Blocknummern stehen noch im Verzeichnis (das RECLAIM-Kommando von POWER/DIENST ersetzt E5H durch 00H, das heißt, das File ist unter user 0 wieder verfügbar, falls die BDOS-Blöcke inzwischen nicht anderweitig benutzt wurden).

☒ KONTAKT ☏

Akademie der Wissenschaften der DDR, Institut für Informatik und Rechentechnik, Rudower Chaussee 5, Berlin, 1199; Tel. 6 74 24 03

Kleines Lexikon der Mikrorechentechnik

D

wie Diskettenformatierung

Zeichnung: Dahmen

Anwendung des FDC U 8272 D

Dr. Eberhard Böhl
VEB Forschungszentrum Mikroelektronik Dresden

Der Floppy-Disk-Controller U 8272 D wird von vielen Betrieben und Institutionen sowie auch von Amateuren benutzt, um vorhandene Büro- oder Personalcomputer mit einem Diskettenlaufwerk zu erweitern.
Nach der grundlegenden Beschreibung des Schaltkreises /1/ und /2/ sollen hier noch einige anwendungsspezifische Hinweise gegeben werden.

Ein zusätzliches Aufzeichnungsformat

Gegenüber den international vergleichbaren Schaltkreistypen μPD 765 A (NEC) und 8272 A (Intel) verfügen der U 8272 D08 und der U 8272 D04 über ein zusätzliches Aufzeichnungsformat mit 128 Byte/Sektor bei doppelter Datendichte. Alle Schaltkreise aus der Produktion des VEB Mikroelektronik „Karl Marx" Erfurt (Produktionsbetrieb) unterstützen dieses Format; Schaltkreise aus der Produktion des VEB Forschungszentrum Mikroelektronik Dresden (Entwicklungsbetrieb) mit dieser Eigenschaft sind mit einem farbigen Punkt gekennzeichnet. Das Format 128 Byte/Sektor MFM besitzt in der DDR und im RGW große Bedeutung. Die damit in Verbindung stehende umfangreiche Software kann auch beim Einsatz des U 8272 weiterhin genutzt werden.
Bei Verwendung dieses Formates ist in den entsprechenden Befehlen $N = 0$ zu setzen (siehe Beschreibung der Befehle in /1/).
Zu beachten ist, daß für doppelte Datendichte eine variable Datenlänge nicht möglich ist; die in den Lese- und Schreibbefehlen angegebene Größe DTL (Datenlänge) hat nur für einfache Datendichte bei 128 Byte/Sektor Bedeutung.

Vorgaben für die Spuraufteilung

Für das Format 128 Byte/Sektor MFM existieren keine Vorgaben bezüglich der Zahl der Sektoren pro Spur und der Länge der Lücke zwischen den Sektoren. Hier müssen von den Anwendern selbst Erfahrungen gesammelt werden, bis sich eine bewährte Spuraufteilung durchsetzt. Um den Umgang mit dem neuen Datenformat zu erleichtern, sollen hier ein paar Hinweise gegeben werden. Für Minifloppy-Laufwerke soll dabei von einer Umdrehungsgeschwindigkeit von $U = 300\ \text{min}^{-1}$ mit ±2% Toleranz ausgegangen werden. Bei doppelter Datendichte sind bei einem Takt von $f_c = 4$ MHz und einem Schreibtakt von $f_{WRC} = 0{,}5$ MHz (Bitzelle 4 μs) folgende Datenbytes nominell auf der Spur verfügbar:

$$Z_n = \frac{f_{WRC}}{2 \cdot U} \text{ Bit/Spur}$$

$$= \frac{f_{WRC}}{2 \cdot U \cdot 8} \text{ Byte/Spur}$$

Z_n = 6250 Byte/Spur.

Bei einer Schwankung der Umdrehungsgeschwindigkeit von ±2% erhält man:

Z_n = (6250 ± 125) Byte/Spur.

– Lückenlänge
Für die Länge der Lücke zwischen den einzelnen Sektoren (Gap 3) muß man unterscheiden zwischen Formatierungs- und Lese-Schreibbefehlen, da es durch die Toleranzen der Umdrehungsgeschwindigkeit insbesondere beim Schreiben sonst zu Komplikationen kommen kann (Überschreiben der Daten des nächsten Sektors). Für das Lesen kann prinzipiell mit einem beliebig kleinen Wert für GPL_1 gearbeitet werden, weil der U 8272 nach dem Verarbeiten eines Sektors nur soviel Bytes, wie in GPL_1 angegeben werden, abwartet, bis er mit der Suche des nächsten Synchronfeldes beginnt.
Zu beachten ist, daß $GPL_1 \geqq 1$ gewählt wird, da bei $GPL_1 = 0$ eine Lücke von 256 Byte angenommen wird. Wird $GPL_1 = 1$ gewählt, so beginnt die Synchronfeldsuche des nächsten Sektors schon nach wenigen Bytes der Gap 3. Ein eventuell auftretender Phasensprung zwischen zwei Sektoren, der durch das Überschreiben des ersten Sektors bei einer anderen Umdrehungsgeschwindigkeit als beim Formatieren entstanden sein kann, bringt den U 8272 nicht aus dem Konzept, da er solange sucht, bis er ein Byte des Synchronfeldes zuverlässig erkannt hat. Auch die Datenseparatorschaltung sollte so beschaffen sein, daß sie ein Aktivieren vor dem Phasensprung mit VCO = H nicht stört. Für den in diesem Beitrag angegebenen digitalen Datenseparator ist das offensichtlich der Fall, weil er sich mit dem nächsten Datenimpuls nach dem Phasensprung sofort wieder synchronisiert.
Bei Schreibbefehlen mit $GPL_1 = 1$ kann es durch die Umdrehungsgeschwindigkeit an der unteren Toleranzgrenze dazu kommen, daß Reste des überschriebenen Sektors nicht mit gelöscht werden. Das führt in der Regel nicht zu Komplikationen, weil nach den Daten und den zwei CRC-Bytes immer mindestens 3 Byte der Gap 3 geschrieben werden. Zu empfehlen ist aber, daß für ein völliges Überschreiben des Sektors gesorgt wird. Bei 128 Byte/Sektor und ±2% muß man mit 6 Byte Differenz rechnen. Für die Lücke ist im Schreibbefehl deshalb immer $GPL_1 \geqq 4$ (besser: $GPL_1 \geqq 6$) anzugeben.
Bei einem Schreiben über mehrere Sektoren mit einem Schreibbefehl wird die Lage der Sektoren zueinander gegenüber der Formatierung nicht geändert. Nach dem Abschluß eines Sektors sucht der U 8272 jeweils wieder nach dem Synchronfeld und der Adreßmarke, sucht die nächste Sektornummer (um 1 erhöht) und schreibt die nächsten Daten nach der Datenmarke in das dazugehörige Datenfeld. Durch diese Verfahrensweise wird verhindert, daß sich die Toleranzen in der Umdrehungsgeschwindigkeit über mehrere Sektoren addieren.

– Anzahl der Sektoren pro Spur
Für die Anzahl der Sektoren pro Spur muß man in Betracht ziehen, daß der U 8272 bei MFM folgende Bytezahlen benötigt:

Index-Vorspann:	146 Byte	
je Sektor:		
Identifikationsfeld	60 Byte	* n Sektoren
Daten (128 Byte/Sektor)	128 Byte	
CRC	2 Byte	
GPL_2	X Byte	

$Z_F = 146 + n\,(190 + x)$.

Für den Wert von GPL_2 im Formatierungsbefehl ist neben den Laufwerkstoleranzen zu beachten, daß der U 8272 beim Lesen und Schreiben zwischen zwei Sektoren etwas Zeit benötigt, um den weiteren Befehlsablauf zu organisieren, bis er das nächste Synchronfeld verarbeiten kann. Für 128 und 256 Byte pro Sektor sind dazu mindestens 3 Byte und die in GPL_1 des Lese-/Schreibbefehls angegebenen Bytes erforderlich; für 512 Byte pro Sektor und mehr ist noch ein weiteres Byte notwendig. Außerdem benötigt der U 8272 (unabhängig davon, was die externe Datenseparatorschaltung zum Einschwingen benötigt) bis zu 5 Byte Synchronfeld; davon müssen die letzten drei Byte unverfälscht bei stabilen RDD- und DW-Signalen anliegen. Mit der Gestaltung des Synchronfeldes wird schon beim Formatieren diese letzte Bedingung erfüllt – es muß nur gewährleistet werden, daß das Synchronfeld beim Schreiben nicht teilweise verkürzt wird oder erst zu spät gelesen werden kann.
Für die erforderliche Lückenlänge muß man demnach bei 128 Byte pro Sektor ansetzen:

* Schreib-/Lesetoleranz pro Sektor
  128 Byte · 4% = 5,12 Byte ≈ 6 Byte
* Anzahl der Byte zur Organisation des Ablaufs zwischen zwei Sektoren + 3 Byte
* programmierte Lückenlänge bei R/W-Befehlen + GPL_1 Byte

Für $GPL_1 = 6$ ergibt sich damit die Mindestlückenlänge in Formatierungsbefehlen zu $X = 15$. Mit diesem Wert erhält man für Minidisketten die Möglichkeit, 29 Sektoren auf einer Spur unterzubringen.

$Z_F = 146 + 29\,(190 + 15)$
$Z_F = 6091$

Der Wert Z_F liegt unterhalb der nominellen Byteanzahl einschließlich der Toleranzen:

$Z_F \leqq Z_n/\text{minimal} = 6125$

Vergleicht man die Werte mit den Empfehlungen für andere Formate (siehe /1/, /2/), so stellt man fest, daß die ermittelte GPL_2 noch größer ist als die GPL_2 einer Spuraufteilung für 256 Byte pro Sektor, während GPL_1 kleiner ist. Für eine andere Spuraufteilung mit 256 Byte pro Sektor MFM werden für $GPL_2 = 50$ und $GPL_1 = 32$ angegeben. Das sind Werte, die die hier ermittelten Mindestwerte für das neue Format noch wesentlich übersteigen. Für das erstgenannte Vergleichsformat muß angenommen werden, daß entweder ein Laufwerk mit noch geringeren Schwankungen der Umdrehungsgeschwindigkeit zugrunde gelegt oder bewußt das Überschreiben bzw. Ignorieren eines Teiles des folgenden Synchronfeldes in Kauf genommen wird, da nicht alle 12 Byte benötigt werden.
Vergleicht man weiterhin die Anzahl der Sektoren pro Spur von 4096 Byte/Sektor bis 256 Byte/Sektor, so stellt man eine Verdopplung bei Halbierung der Byteanzahl fest. Auf 128 Byte/Sektor extrapoliert erhält man 32 Sektoren, die aber selbst ohne Lücke zwischen den Sektoren nicht auf einer Spur Platz finden würden.
Dem Versuch, mehr als 29 Sektoren auf einer Spur unterzubringen (was theoretisch noch möglich wäre), steht eine geringere Datensicherheit und größere Fehleranfälligkeit gegenüber. Selbst die hier als Mindestwerte ermittelten Größen sind nur als Richtwerte zu betrachten, die erst in der Praxis erprobt und gegebenenfalls noch verändert werden müssen. Die Betrachtungen in diesem Abschnitt können dazu als Hilfestellung dienen.

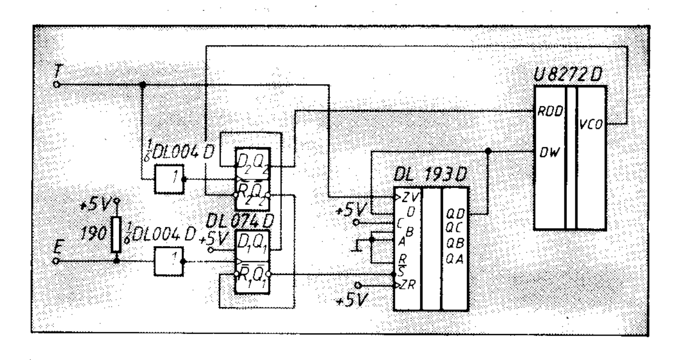


Bild 1 Digitaler Datenseparator für MFM, Eingangssignale T: Takt, E: Lesedaten (negiert)

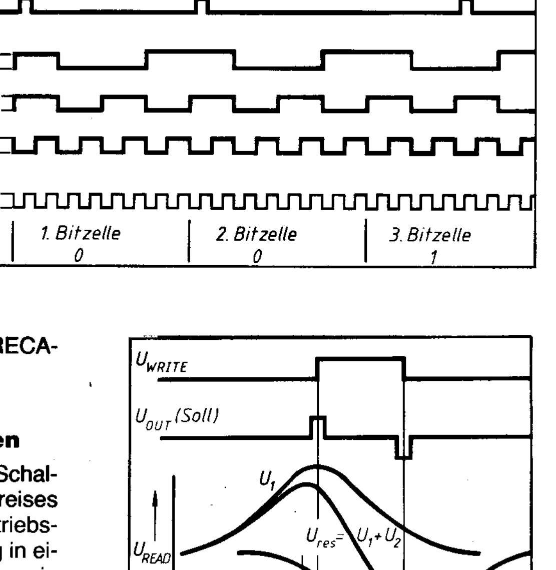


Bild 2 Taktdiagramm für MFM, (Schaltung: Bild 1)

Für Normaldisketten sind prinzipiell die gleichen Vorgaben für die Lückenlänge verwendbar, wenn die Laufwerke höchstens die gleichen Toleranzen in der Umdrehungsgeschwindigkeit aufweisen.
Mit $GPL_1 = 6$ und $GPL_2 = 15$ sind auf einer Spur 49 Sektoren mit 128 Byte pro Sektor unterzubringen. Zusammen mit dem Indexvorspann wären insgesamt 10191 Byte erforderlich. Bei 360 min^{-1} Umdrehungsgeschwindigkeit mit ±2% Toleranz werden damit die nominell verfügbaren 10416 ± 209 Byte gut ausgenutzt.

Unterstützung durch das Betriebssystem

Für die festzulegende Sektorzahl pro Spur ist sowohl für die Mini- als auch für die Normaldisketten zusätzlich zu den hier betrachteten Gesichtspunkten noch zu beachten, welche Werte die benutzte Software unterstützt.
Für 128 Byte/Sektor MFM kann es sein, daß dann die hier vorgeschlagene Spuraufteilung nicht ausgenutzt werden kann. Umgekehrt gibt es Betriebssysteme, die mehr als die in /1/ und /2/ angegebenen Sektoren pro Spur verarbeiten. Insbesondere sind bei 1024 Byte/Sektor MFM auf einer Minidiskettenspur prinzipiell auch 5 Sektoren unterzubringen, obwohl das nicht den angegebenen Vorzugswerten des U 8272 und dessen Vorbildtypen entspricht. Mit den hier gegebenen Hinweisen ist eine Spuraufteilung mit 100 Byte Lückenlänge zwischen den Sektoren beim Formatieren und 48 Byte bei Lese-/Schreibbefehlen denkbar.

Mehr als 77 Spuren

Der U 8272 D unterstützt wie seine Vergleichstypen Diskettenlaufwerke mit 77 Spuren. Für die Ansteuerung von Geräten mit 80 und mehr Spuren ist zu beachten, daß der Befehl RECALIBRATE nach 77 STEP-Impulsen abgebrochen wird, wenn kein TRACK-0-Signal erkannt wird. Zur ordnungsgemäßen Bewegung des Schreib-Lese-Kopfes auf die äußerste Spur sind gegebenenfalls mehrere RECALIBRATE-Befehle nacheinander erforderlich. Der Abbruch des RECALIBRATE-Befehls wird wie auch die erfolgreiche Beendigung durch das INT-Signal angezeigt. Da der Befehl keine Erzeugnisphase besitzt, ist erst durch einen nachfolgenden SENSE-INTERRUPT-STATUS-Befehl die Information über den ordnungsgemäßen Abschluß erhältlich. Im Statusregister 0 wird in jedem Fall das SEEK-End-Bit (D5) gesetzt, bei Ausbleiben des TRACK-0-Signals zusätzlich noch das Equipment-Check-Bit (D4). Damit kann entschieden werden, ob ein erneuter RECALIBRATE-Befehl erforderlich ist.

Undefiniertes Verhalten nach dem Spannungseinschalten

Der U 8272 D enthält verschiedene Schaltungsteile, die das Testen des Schaltkreises erleichtern. Bei Einschalten der Betriebsspannung kann der Schaltkreis zufällig in einen Zustand des Testmodus gelangen, in dem er nicht wie im Datenblatt beschrieben reagiert. Eine Rückkehr in den normalen Betriebsmodus ist möglich durch das Aktivieren des RESET-Signals in Verbindung mit dem eigentlich nur im Testmodus als Eingang benutzten DRQ-Pin oder mit dem Signal TC.
In /2/ wurde dazu empfohlen, einen Widerstand von 4,7 kΩ zwischen U_{cc} und dem Pin DRQ anzubringen. Dadurch werden alle Einschaltprobleme in Verbindung mit dem Aktivieren des RESET-Signals gelöst.

Digitaler Datenseparator

Zur Trennung des Takt-Daten-Gemisches ist ein Datenseparator erforderlich, für den üblicherweise eine PLL-Schaltung eingesetzt wird.
In /3/ wird neben anderen Schaltungen auch ein digitaler Datenseparator vorgestellt, der sich schon bei verschiedenen Anwendern bewährt hat (Bild 1). Im Bild 2 ist das Taktdiagramm für diese Schaltung bei MFM dargestellt.
Zur prinzipiellen Funktionsweise ist zu sagen, daß mit jedem Eingangsimpuls an E (negierte Lesedaten) der Binärzähler DL 193 D auf den Wert entsprechend dem Dezimaläquivalent 4 oder 12 gesetzt wird. Mit den folgenden Taktimpulsen wird der Zähler weitergezählt, und mit jedem Impuls am Eingang E synchronisiert sich der Zähler neu. Liegen zwei Eingangsimpulse 16 ± 2 Taktimpulse auseinander, so wird der gleiche Datenwert angenommen. Das DW-Signal hat dann die gleiche Polarität. Bei 24 ± 2 Taktimpulsen Abstand zwischen zwei Eingangsimpulsen wird ein Signalwechsel festgestellt, der sich in der anderen Polarität des DW-Signals während des RDD-Signals äußert. Prinzipiell ist die gleiche Schaltung auch für FM verwendbar, wenn man zwischen dem Ausgang Q_D des Zählers und dem DW-Eingang des U 8272 ein Flip-Flop als Frequenzteiler schaltet. Der Paralleleingang des Zählers ist dazu auf den Wert 8 (Dezimaläquivalent) zu setzen, und das zusätzliche Flip-Flop vor dem DW-Eingang muß seinen Zustand während des Ladevorganges beibehalten. Bei zirka 16 Takten Abstand zwischen zwei Eingangsimpulsen liegt dann ein H und bei zirka 32 Takten Abstand ein L vor. Sowohl für den Amateur als auch für industrielle Anwendungen ist die MFM-Anwendung von größerem Interesse, und die Schaltung von Bild 1 hat sich schon in der Praxis bewährt. Es liegt deshalb zunächst kein Grund vor, die in /3/ geäußerten Bedenken bezüglich der Funktionstüchtigkeit und Zuverlässigkeit der angegebenen Schaltung zu teilen, zumal die Ursachen der dort beschriebenen Ausfälle nicht ermittelt werden konnten. Es wäre sinnvoll, diese Schaltung auch im industriellen Einsatz zu erproben.

Bild 3 Verschiebung der Ausgangsspannungszeitpunkte um Δt durch Überlagerung der induzierten Spannungen zweier eng benachbarter Flußwechsel bei MFM

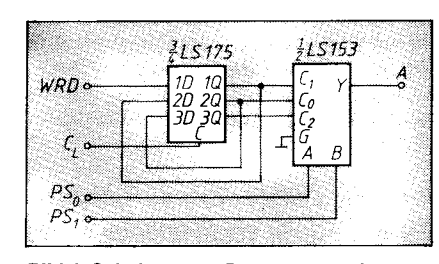


Bild 4 Schaltung zur Precompensation

Precompensation

Bei MFM-Codierung kommt es bedingt durch die hohe physische Datendichte und durch die Art der Aufzeichnung auf der Diskette bei bestimmten Datenkonstellationen durch Überlagerung mehrerer benachbarter Flußwechsel zu einer scheinbaren Verschiebung der Datenbits, die in einer zeitlichen Verän-

derung der Lesespannungsmaxima zum Ausdruck kommt. Berücksichtigt man, daß jede Eins als Wechsel der Magnetisierungsrichtung auf der Diskette abgespeichert wird, so ist offensichtlich, daß dieser Effekt am Anfang und Ende eines Bereiches mit lauter Einsen auftritt. Im Bild 3 werden der Verlauf der Lesespannungen zweier benachbarter Einsen (U_1 und U_2) und die daraus resultierende Gesamtlesespannung U_{res} dargestellt. Deutlich sieht man in diesem Bild die zeitlichen Abweichungen der Maxima der Einzelspannungen von denen der resultierenden Spannung, wodurch sich eine Verschiebung der RDD-Impulse ergeben würde. Um dem zu begegnen, wird schon beim Aufzeichnen auf die Diskette durch eine zeitliche Verschiebung der betreffenden Impulse eine physisch verschobene Anordnung der betreffenden benachbarten Flußwechsel erreicht. Damit entstehen beim Lesen die Maximalwerte der resultierenden Lesespannung zu den erwarteten Zeitpunkten. Der erforderliche Betrag der zeitlichen Verschiebung beim Schreiben ist für verschiedene Laufwerktypen nicht konstant, da es auf die konkreten Bedingungen des Lese-/Schreibkopfes und dessen Umgebung ankommt.
Mit Hilfe der vom U 8272 bereitgestellten Preshift-Signale und einem laufwerksspezifischen Taktsignal C_L läßt sich mit drei D-Flip-Flops und einem 4-zu-1-Multiplexer ganz einfach die erforderliche Schaltung aufbauen. Das WRD-Signal des U 8272 wird dazu an den Eingang der in Reihe geschalteten D-Flip-Flop geschickt, die alle mit dem gleichen Takt C_L getaktet werden. Der Multiplexer schaltet dann für den Fall PS0 = PS1 = L den Ausgang des 2. D-Flip-Flop zum Laufwerk durch, während bei PS0 = H der des 1. und bei PS1 = H der des 3. D-Flip-Flops durchgeschaltet werden. Eine entsprechende Schaltung, wie auch in /3/ angegeben, zeigt Bild 4.

Literatur
/1/ Böhl, E.: Integrierte Floppy-Disk-Controller-Schaltungen U 8272 D 08 und U 8272 D 04, Radio, Ferns., Elektron., Berlin 36 (1987) 11, S. 703
/2/ Böhl, E.: Der Floppy-Disk-Controller U 8272 D und sein Einsatz. Mikroprozessortechnik, Berlin 2 (1988) 4 u. 7, S. 102 u. 200
/3/ Kramer, M.: Praktische Mikrocomputertechnik. Berlin: Militärverlag 1987

✉ KONTAKT ☏
VEB Forschungszentrum Mikroelektronik Dresden, Abt. EAP, Grenzstraße 28, Dresden, 8080; Tel. 58 82 46

Objektorientierung in der Informatik

Frank Kühle
Technische Universität Dresden, Informatik-Zentrum

Dipl.-Ing. Frank Kühle (26) studierte von 1982 bis 1986 an der TU Dresden, Sektion Informationsverarbeitung. Seine Diplomarbeit behandelte den rechnergestützten Softwareentwurf für Systeme paralleler Prozesse. Er beschäftigt sich zur Zeit mit der Weiterentwicklung des aus seinem Wissenschaftsbereich hervorgegangenen Systems COSEM zu einer integrierten, objektorientierten Entwicklungsumgebung zur flexiblen Computerstützung von Softwareentwurfsprozessen.

Getragen durch verschiedene Systeme und Sprachen wie Smalltalk (Goldberg, Kay), ACT-1 (Hewitt) oder KRL ist aus der Entwicklung der Informatik das sogenannte objektorientierte Modell der Informationsverarbeitung hervorgegangen. Die darauf basierende objektorientierte Programmierung wurde mit Smalltalk bekannt und wird heute in sehr differenzierter Weise auch von verschiedensten objektorientierten Erweiterungen bekannter Programmiersprachen unterstützt. Der Grad der Objektorientierung wird nach einem Vorschlag von B. Meyer in 7 Niveaus abgestuft. Es läßt sich aber zeigen, daß die objektorientierte Programmierung in Vervollkommnung der bekannten Prinzipien Modularisierung/Datenkapselung und Datenabstraktion entstand und daraus folgerichtig und notwendig 4 Paradigmen hervorgehen, die von einer objektorientierten Programmiersprache unterstützt werden sollten.

„Instead of a bit-grinding processor raping and plundering data structures, we have a universe of wellbehaved objects that courteously ask each other to carry out their various desires."
D. H. H. Ingalls /1/

Vor einigen Jahren wurden objektorientierte Programmiersysteme vor allem wegen ihrer bemerkenswert angenehmen Nutzerinterfaces bekannt, namentlich durch das der Systeme Smalltalk-76 und Smalltalk-80 /2/. Zu deren Grundkonzeption gehört ein vollgrafikfähiges Display, auf dem sich, verwaltet durch Fenstertechnik, die Objekte in einer ihrer externen Sichten darstellen. Objektorientierung ist aber nicht durch irgendeine besondere Form der Repräsentation von Objekten gegeben und auch nicht dadurch, daß man vorher übliche und anerkannte Begriffe wie Prozedur, Variable, Task oder Mailbox einfach in Objekte verwandelt. Derlei Wortspielerei ist zuweilen zu beobachten und dient mitunter sehr vordergründig dem Ziel, sich das hochmodische Attribut *objektorientiert* zu verleihen. Allerdings wird einer falschen Benutzung des Begriffs durch seine Unschärfe und scheinbare Unverfänglichkeit Vorschub geleistet. Immerhin kann es niemandem untersagt werden, ihn für ein Konzept oder System zu benutzen, das trivial objektorientiert im eigentlichen Sinn der Wortzusammensetzung ist. Jedoch ist eine einigermaßen fest umrissene Bedeutung aus der Entwicklung der Informatik hervorgegangen und sollte respektiert werden.

Was ist objektorientierte Programmierung (OOP)?

□ Zur Realisierung einer Informationsverarbeitungsaufgabe mit dem Computer werden **Objekte** mit speziellen Eigenschaften und einem speziellen Verhalten entworfen. Dabei werden gleichartige Objekte zu einer Klasse zusammengefaßt.
□ Objekte werden als **Instanz einer Klasse** erzeugt und existieren dann eine gewisse Zeitdauer.
□ Als Ausdruck der Individualität der Objekte können ihre Eigenschaften unterschiedlichste Ausprägungen haben und sich im Verlaufe ihrer Existenz ändern. Das heißt, jedes Objekt hat einen eigenen, nur **lokal sichtbaren Zustand.**
□ Das Verhalten eines Objekts repräsentiert sich in seinen individuellen **Reaktionen beim Empfang einer Nachricht.** Aktiviert durch eine Nachricht kann ein Objekt seinerseits ein oder mehrere Nachrichten an andere Objekte senden und/oder seinen Zustand ändern und/oder dem Absender der Nachricht eine Antwort geben.
□ Die richtigen Reaktionen werden durch das Objekt bestimmt, genauer durch seine Klasse, und sind in einer Menge von **Methoden** beschrieben, die jeweils an eine Klasse gebunden sind. Wesentlich ist, daß ein und dieselbe Nachricht bei Objekten unterschiedlicher Klassen verschiedene Reaktionen hervorrufen kann (Polymorphismus).
□ Die einmal für eine Klasse beschriebenen Eigenschaften und Verhaltensweisen (Methoden) sind jeder Instanz einer Klasse eigen.
□ Klassen können als **Spezialisierung** (Subklasse) einer anderen Klasse (Superklasse) definiert werden und erben dabei deren Eigenschaften und Methoden, wenn sie nicht durch eigene ersetzt werden. Es entsteht eine **Klassenhierarchie,** in der Mechanismen der **Vererbung** wirken.
□ Der gesamte Informationsverarbeitungsprozeß wird als **Nachrichtenaustausch** aufgefaßt.
□ Man programmiert also, indem man Klassenhierarchien entwirft und konstante Objekte erzeugt und benennt. Das Programm wird aktiviert, indem man eine oder mehrere Nachrichten an bestimmte Objekte sendet. Danach läuft ein Nachrichtenaustausch ab, in dessen Verlauf Objekte erzeugt werden, sich deren Zustände ändern und Objekte vernichtet werden.

In der Tat gibt es noch sehr differenzierte Auffassungen von dem, was die fundamentalen Prinzipien der Objektorientierung wirklich sind. Es wurde hier versucht, einen ausgewogenen Durchschnitt der in der Literatur zum **objektorientierten Modell der Informationsverarbeitung** zu findenden Aussagen zusammenzufassen. Ein derartiges Modell kann nicht nur objektorientierten Programmiersprachen, sondern auch anderen Systemen der automatisierten Informationsverarbeitung auf unterschiedlichen Betrachtungsebenen zugrunde liegen. Beispielsweise gibt es Objektorientierung in Betriebssystemen, Simulationssystemen und auch in der Gestaltung und Implementierung von grafischen Nutzerinterfaces. Über das hier dargelegte Modell hinaus findet man in verschiedenen Systemen weitere Verfeinerungen und Differenzierungen hinsichtlich der **Vererbungsdisziplinen,** der vereinheitlichten Auffassung der Objekte (Klassenobjekte und Instanzenobjekte nicht unterschieden) oder der **Kombination von Methoden** /13/.

Man kann die Entwicklungslinien der objektorientierten Programmierung wie folgt skizzieren /4/, /5/. Seit 1969 entwickelte *Alan Kay* Ideen zu einem flexiblen elektronischen Denkhilfsmittel, dem **Dynabook.** Zur Erreichung einer hohen Flexibilität durch leichte Änderbarkeit aller Systemteile durch den Benutzer griff er das Klassenkonzept von

SIMULA auf und bereicherte es um eine uniforme Kontrollstruktur, den Nachrichtenaustausch. Diese Kombination nannte er „Objekt-orientierte Programmierung". Damit wurde von A. Kay der Grundstein für die Entwicklung von **Smalltalk**-Systemen (in der Learning Research Group am Xerox Palo Alto Research Center) gelegt, mit denen der Objektbegriff, die Klassenbildung und der Nachrichtenaustausch und damit die objektorientierte Programmierung in immer qualifizierterer Weise realisiert wurden /2/. Dabei ist Smalltalk gewissermaßen der Softwareteil von Alan Kays „personal computing vision", dem Dynabook. Ziel der Learning Research Group war es, die individuellen Fähigkeiten zur kreativen Nutzung des Dynabook zu unterstützen /4/. Das Flex-System, Smalltalk-72, -74 und -76 erweiterten die objektorientierte Betrachtungsweise auf mehr und mehr Elemente einer Programmiersprache und ihrer Programmierumgebung. Beispielsweise waren in Smalltalk-72 die Arithmetik, Listen- und Kontrollstrukturen durch Objekte und Nachrichten repräsentiert, Klassen noch nicht. Sie wurden in Smalltalk-74 als Objekte eingeführt. Das Smalltalk-76-System erlaubte, Relationen zwischen Klassen zu beschreiben (Vererbung) und erweiterte die Objektorientierung auch auf das Nutzerinterface. Smalltalk-80 ist die letzte bekanntgewordene Version mit einer weitgehenden Verallgemeinerung der zugrundeliegenden Architekturprinzipien /1/.

Neben Smalltalk wurde eine weitere Entwicklungslinie von *C. Hewitt* vorangetrieben. Er griff etwa 1972 die von Kay entwickelten Ideen zur Realisierung intelligenter Problemlöser auf und fügte sie in sein **Actor-Modell der Informationsverarbeitung** ein. Dabei arbeitete Hewitt die potentielle Parallelität der kommunizierenden Objekte heraus. Dieser besondere Aspekt der OOP wird heute vielfach herausgegriffen und auf die außerordentlich aktuelle und stark in der Forschung begriffene Programmierung paralleler Prozesse angewendet /6/. Auch die von Hewitt verfolgte Entwicklungslinie mündete in Programmiersprachen und Systeme, namentlich in Versionen von PLANNER, PLASMA und ACT-1. Eine Reihe von Arbeiten in der Künstlichen Intelligenz zur **Theorie der Frames** (Minsky) und deren Implementation in Wissensrepräsentationssprachen wie KRL, KEE, FRL oder UNITS sowie Arbeiten zu abstrakten Datentypen (Parnas), sind ebenfalls mit der Herausbildung der Objektorientierung verbunden.

Anfang der achtziger Jahre entstand ein verstärktes Interesse an der objektorientierten Programmierung. Die aus der Entwicklung hervorgegangenen Resultate und Erfahrungen nahmen einen nicht unerheblichen Einfluß auf **Sprachentwicklungen.** Neben der Schaffung gänzlich neuer objektorientierter Programmiersprachen, da gab es sehr wenige, z. B. EIFFEL /7/, wurden hauptsächlich **Erweiterungen bekannter Sprachen** um Klassenbildung, Nachrichtenaustausch und die damit verbundenen Mechanismen vorgenommen, z. B. Objekt Pascal, Objective C, Exper Common LISP und auch Object Assembler /8/. Wie sich zeigen wird, muß man das Maß der Objektorientierung in solchen Spracherweiterungen sehr differenziert betrachten.

In der Menge von Programmiersprachen, die eine objektorientierte Ergänzung erfahren haben, nimmt **Lisp** eine besondere Stellung ein, z. B. ObjTalk /9/, MAIL /10/, Common Talk, LOOPS usw. für die Lisp-Maschine des MIT entwickelte Zeta-Lisp wurde gleich mit einer objektorientierten Erweiterung, den **Flavors** bekannt /11/. Eine solche Erweiterung der Lisp-Systemfunktionen (Aufsetzen einer Schicht von Funktionen) ist ureigenstes Prinzip der Lisp-Programmierung. Sie erweist sich hier als besonders einfach und naheliegend, da sich das Erzeugen und Vernichten von Objekten, der Nachrichtenaustausch und die Vererbung sowie die inkrementelle Erweiterung von Klassen und Methoden in interaktiver Arbeit organisch in eine Sprache mit automatischer Speicherverwaltung, dynamischer Bindung von Programmobjekten, einem universellen, im Prinzip einzigen Datentyp „Atom" und uniformer Repräsentation von Daten und Programmen einordnen läßt. Die uniforme Repräsentation der Lisp-Programme in Verbindung mit der dynamischen Auswertung ermöglicht sogar die nutzerorientierte Definition von Steuerflußstrukturen (Steuerabstraktion). Damit lassen sich selbst die Steuerstrukturen in Methodendefinitionen spezifisch objektorientiert realisieren /12/.

Das Lisp-eigene Prinzip der Programmentwicklung als sukzessive Erweiterung eines Kerns von Systemfunktionen in interaktiver Arbeitsweise wird durch die flexible Erweiterbarkeit eines objektorientierten Programms vervollkommnet (Hinzufügen von Methoden oder Klassen erfordert keinerlei Veränderungen am Bestehenden). Andererseits können Konzepte und Zielstellungen der Objektorientierung, zum Beispiel eine explorative Arbeitsweise, in Lisp in hoher Vollendung realisiert werden. Infolge dieser fruchtbaren Symbiose ist die **objektorientierte Programmierung in Lisp** zu einem relativ eigenständigen Gebiet avanciert /10/, /13/, /14/. Verbunden mit der Objektorientierung werden derartige Lisp-Erweiterungen vielfach zu Wissensrepräsentationssystemen entwickelt.

In der Vielzahl bereits existierender Sprachen und Systeme, die das Attribut *objektorientiert* beanspruchen, gestaltet sich eine Endgültige Klassifizierung schwierig. Viele Systeme sind zumindest in Ansätzen objektorientiert, wenn man Datenmoduln, abstrakte Datentypen oder ähnliches bilden kann, da der Klassenbegriff als Fortentwicklung des Datentypbegriffs verstanden werden kann. Auch die Entwicklung der bekanntesten objektorientierten Programmiersprache Smalltalk stellt ein gewisses Kontinuum in der Vervollkommnung der OOP dar. Für den Grad der Objektorientiertheit werden von B. Meyer /15/ 7 Niveaus unter der Überschrift „Steps towards object orientedness" genannt.

[1] **Objekte als Moduln** realisiert – Systeme werden auf der Basis ihrer Datenstrukturen modularisiert
[2] **Datenabstraktion** – Objekte als Implementationen abstrakter Datenstrukturen
[3] **Automatische Speicherverwaltung** – dynamisches Erzeugen und Vernichten von Objekten
[4] Sprachkonstrukt *Klasse* vereinigt den Modul- und Typenaspekt
[5] **Vererbung** – eine Klasse kann als eine Erweiterung oder Einschränkung einer anderen Klasse definiert werden
[6] **Polymorphismus** – Programmeinheiten können auf Objekte von mehr als einer Klasse zugreifen, und Operationen können in verschiedenen Klassen verschiedene Realisierungen haben
[7] **multiple Vererbung** – eine Klasse kann Erbe von mehreren Klassen sein

Diese von Meyer definierten Niveaus der Objektorientiertheit stellen offensichtlich mehr die Entwicklungsgeschichte der objektorientierten Programmierung als ein flexibles Modell dar, aus dem jeder sich sein ganz persönliches Maß der Objektorientierung herausnehmen kann. Es läßt sich nämlich zeigen, daß die objektorientierte Programmierung in Vervollkommnung der aus der Programmierungstechnik bekannten Prinzipien Modularisierung/Datenkapselung und Datenabstraktion entstand (vgl. auch /16/) und daraus folgerichtig und notwendig 4 Paradigmen hervorgehen, die von einer objektorientierten Programmierpache unterstützt werden müssen, da sonst der beabsichtigte Gewinn nicht erreicht wird.

Was darf man von objektorientierten Programmiersprachen fordern?

G. A. Pascoe beschreibt in /17/ 4 Elemente, die eine Sprache notwendig aufweisen muß, um die objektorientierte Programmierung unterstützen zu können.

① **information hiding**	(Verkapselung, Lokalität)
② **data abstraction**	(Datenabstraktion)
③ **dynamic polymorphism** (dynamic binding)	(dynamischer Polymorphismus)
④ **inheritance**	(Vererbung)

Ausgehend von seinen Erläuterungen, teilweise bezugnehmend auf Modula-2 und Ada, läßt sich die genannte Entwicklung in der Programmierungstechnik sehr anschaulich machen. Dabei ist die Programmiersprache in ihrer Einheit mit dem sprachverarbeitenden System und seinen Werkzeugen zu sehen, denn Fortschritte in der Effektivität der Softwareentwicklung werden gerade auch von der besseren Unterstützung des Entwerfens und Programmierens als Prozeß erwartet.

TERMINE

Anwendersymposium zur Vorstellung des RVS K 1840 für potentielle Nutzer

WER? WGMA
WANN? Ende März 1989
WO? Leipzig
WAS? Darstellung der Eigenschaften des Systems zur effektiven Vorbereitung des Einsatzes.
Schwerpunkte:
- Konzeption und Konfiguration der Hardware
- Betriebssysteme SVP 1800 und MUTOS 1800
- Compiler und Standardsoftware
- Einsatzbedingungen

WIE? Meldungen an: Wissenschaftlich-Technische Gesellschaft für Meß- und Automatisierungstechnik, Clara-Zetkin-Str. 115/117, Berlin, 1086; Tel. 2 26 52 18 oder 2 26 53 04

Müller

Das Information hiding (Verkapselung, Lokalität) hat die Reduzierung und strenge Reglementierung von Abhängigkeiten zwischen Softwarekomponenten (z. B. Moduln) und damit die Erhöhung von Zuverlässigkeit und Änderbarkeit zum Ziel. Das heißt, der aktuelle Zustand ist stets nur innerhalb der betreffenden Komponente sichtbar und kann von außen nur über eine wohldefinierte Schnittstelle erfragt oder beeinflußt werden. In der OOP stellen sich Wechselwirkungen als Nachrichtenaustausch zwischen Objekten als Softwarekomponenten dar. Es werden von der Sprache entsprechende Mittel (z. B. Moduln) zur Implementierung von Objekten gefordert, deren lokale Instanzenvariablen den aktuellen Zustand enthalten. Sie dürfen sich nur über eine bestimmte Menge von Nachrichten (z. B. die exportierten Prozeduren) beeinflussen lassen.

In der Programmierungstechnik ist es allgemeines Prinzip, komplexe Datenstrukturen mittels einfacherer Datentypen zu implementieren und die zugehörigen Operationen auf elementarere der einfachen Datentypen zurückzuführen. **Data abstraction** (Datenabstraktion) ist eine Qualifizierung dieses allgemeinen Prinzips, da von der speziellen Implementierung des abstrakten Datentyps (beispielsweise Stack oder Queue) tatsächlich abstrahiert werden kann. Eine wohldefinierte Schnittstelle, beispielsweise eine Menge von Prozeduren, beschreibt das Verhalten der abstrakten Datenobjekte vollständig. Derartige Datenabstraktionsmechanismen muß eine die OOP unterstützende Sprache besitzen, um die Definition von Klassen mit ihren Schnittstellen in Gestalt von Methoden und das Erzeugen von Objekten zu gestatten. Für die Klassen Stack (Kellerspeicher) und Queue (Warteschlange) handelt es sich z. B. um die Methoden *Initialize, Push, Pop.*

Das bis hierhin erläuterte Niveau von *information hiding* und *data abstraction* ist in Modula-2 und Ada gewährleistet und in modernen höheren Programmiersprachen üblich. Jedoch wird die mit der OOP verfolgte Weiterentwicklung des Datenabstraktionsprinzips durch keine konventionelle Programmiersprache innerhalb ihres Datentypkonzepts vollständig unterstützt.

Der Nachrichtenaustausch ist die adäquate Kontrollstruktur einer vervollkommneten Datenabstraktion. Jedes Objekt ist selbst für die zweckmäßige Reaktion auf eine empfangene Nachricht verantwortlich. Die Prozeduren werden in ihrer zentralen Stellung („Ich habe eine Prozedur, worauf kann ich sie anwenden und mit welchen Ergebnissen?") durch Klassen abgelöst („Welches Verhalten hat diese Menge von Objekten, welche Nachrichten können empfangen werden?"). Folglich werden Objekte verschiedener Klassen auf ein und dieselbe Nachricht mit unterschiedlichen Methoden reagieren. Dies ist, wie sich im folgenden zeigen wird, letztlich nur dynamisch entscheidbar, also **dynamic polymorphism.**

Für die beiden Klassen Stack und Queue mögen jeweils die Methoden Initialize, Push und Pop definiert sein. Empfängt ein Stack- oder Queue-Objekt die Nachricht mit der Aufforderung zu seiner Initialisierung, wird vom Objekt je nach Klassenzugehörigkeit seine spezielle Methode benutzt. In Modula-2 ist die gleichzeitige Benutzung zweier gleichnamiger Prozeduren jedoch problematisch. Man muß entweder unterschiedliche Bezeichnung der exportierten Prozeduren in allen abstrakten Datentypen garantieren oder ihre Benutzung qualifizieren, das heißt, den Datentyp dem Prozedurnamen voranstellen. Beides würde dem Wesen der Objektorientierung widersprechen. Ada gestattet, das Problem teilweise durch Operatorüberladung zu lösen. Dabei wird zur Compilezeit der richtige Operator durch Analyse der Operanden bestimmt und eingebunden. Zur Erreichung flexibler Erweiterbarkeit und Änderbarkeit muß in der OOP die Klassenzugehörigkeit der „Operanden" jedoch variabel sein und kann erst zur Laufzeit bestimmt werden. Die Realisierung dieses dynamischen Polymorphismus fordert von einer die OOP unterstützenden Sprache dynamische Bindemechanismen bzw. eine dynamische Typprüfung.

Eine weitere Qualifizierung der Datenabstraktionsmechanismen, die die OOP anbietet, ist die Möglichkeit, Klassen als Spezialisierung anderer Klassen zu definieren. Mit der Zuordnung zu einer Superklasse geht das Ererben allgemeiner Verhaltensweisen und Eigenschaften unter Hinzufügen speziellerer einher. Da die Prozeduren in der objektorientierten Betrachtungsweise als Methoden an die Klassen gebunden sind und das Verhalten der Instanzen dieser Klassen beschreiben, ist **inheritance** (Vererbung) eine natürliche Folgerung, um ähnliche oder gleiche Verhaltensweisen in die Beschreibung spezialisierter Klassen zu übernehmen und gegebenenfalls noch durch Methodenkombination abzuwandeln. Die Vererbung erlaubt eine sehr gute Mehrfachnutzung von „Kode", ohne undurchschaubare, verhängnisvolle Abhängigkeiten zuzulassen und steigert damit die Qualität der Software durch bessere Faktorisierung. Ein derartiger Mechanismus der Spezialisierung wird innerhalb des Datentypkonzepts herkömmlicher Programmiersprachen nicht unterstützt.

Die Diskussion der vier von einer objektorientierten Programmiersprache zu fordernden Charakteristika ging davon aus, daß die Objektorientierung in der Programmierung als Vervollkommnung von Datenabstraktionsmechanismen verstanden wird und demzufolge innerhalb des Wertart- bzw. Datentypbegriffs interpretiert werden muß. Wird allerdings ein einziger universeller Datentyp *Objekt* definiert und die dynamische Bindung und Typprüfung explizit programmiert, so kann man Objektorientierung in Sprachen mit üblichen Datenabstraktionsmechanismen simulieren.

Was darf man von einem objektorientierten Programmiersystem erwarten?

Die Arbeitsweise eines objektorientierten Programmiersystems betreffend soll hier nur eine einzige wesentliche Frage diskutiert werden, die einen grundlegenden Einfluß auf die **Art und Weise der Programmiertätigkeit** hat. Die OOP bringt mit der Bindung der Methoden an Klassen als Beschreibung der spezifischen Reaktionen auf empfangene Nachrichten, eingeschlossen die Möglichkeit, daß ein und dieselbe Nachricht bei Objekten verschiedener Klassen unterschiedliche Reaktionen hervorrufen kann, bedeutende Vorteile für die Erweiterbarkeit und Änderbarkeit der Programme. Soll eine neu hinzukommende Subklasse anders auf bestimmte Nachrichten oder auch auf andere Nachrichten reagieren als die Superklasse, so müssen stets nur lokal und ohne Änderung bestehender Programmteile Ergänzungen vorgenommen werden. Die Erweiterbarkeit schon während des Softwareentwicklungsprozesses auszunutzen und diesen damit **inkrementell** und **explorativ** (erkundend, erforschend) zu gestalten, ist nach Auffassung vieler Wissenschaftler erklärtes Ziel der objektorientierten Programmierung und von ihr nicht zu trennen.

Die Arbeit mit Klassen läßt eine Programmierung durch schrittweise Erweiterungen der Klassenhierarchie und der Beschreibung des Verhaltens von Mengen gleichartiger Objekte im Dialog mit dem Computer besonders geeignet erscheinen. Dafür liefert die einfache Erweiterbarkeit objektorientierter Entwürfe, ohne bestehende Programmteile ändern zu müssen, die Voraussetzungen. Außerdem wird der Zugang zu den im allgemeinen ungewohnten und schlecht akzeptierten abstrakten Datentypen erleichtert. Hier wird ersichtlich, daß qualitativ bessere Softwareentwicklung Anforderungen an die Programmiersprache wie auch an das sprachverarbeitende System stellt. Man kann noch deutlicher sagen, daß die objektorientierte Programmierung eine Programmiertätigkeit herausfordert, in der **lange Edit-compile-link-test-Schritte** keinen Platz mehr haben. Im Prozeß der Vereinheitlichung der Auffassungen von der OOP scheint sich die notwendige Voraussetzung *explorative Programmierung* mehr und mehr durchzusetzen. Damit würden die heute noch als objektorientiert bezeichneten Sprachen/Sprachsysteme wie Object Pascal, Object Assembler und Objective C zugunsten von Smalltalk, Neon, Exper Common Lisp, Objtalk, Object Logo, Zeta Lisp, EIFFEL usw. in den Hintergrund treten.

Literatur

/1/ Ingalls, D. H. H.: Design Principles Behind Smalltalk. Byte 6 (1981) 8, S. 286

/2/ The Smalltalk-80 System, Learning Research Group of Xerox Palo Alto Research Center. Byte 6 (1987) 8, S. 36

/3/ Stoyan, H.; Görz, G.: Was ist objektorientierte Programmierung? In: /5/, S. 9

/4/ Goldberg, A.: Introducing the Smalltalk-80 System. Byte 6 (1981) 8, S. 14

/5/ Stoyan, H.; Wedekind, H. (Hrsg.): Objektorientierte Software- und Hardwarearchitekturen, 5.–6. Mai 1983, Berlin (West). Berichte des German Chapter of the ACM 15 (1983), Stuttgart: Teubner 1983

/6/ Yonezawa, A.; Loeper, H.; Jäkel, H.-J.: The Rendezvous Concept – a Programming Tool for Parallel Processing. EIK 21 (1985) 9, S. 429

/7/ Meyer, B.; Nerson, J. M.; Matsuo, M.: EIFFEL: Object-Oriented Design for Software Engineering. Tagungsbericht, First European Software Engineering Conference, S. 237

/8/ Schmucker, K. J.: Object-Oriented Languages for the Macintosh. Byte 11 (1987) 8, S. 177

/9/ Lemke, A. C.: ObjTalk 84 Reference Manual. Dept. of Comp. Sc., University of Colorado, Boulder, Technical Report CU-CS-291-85, June 1985

/10/ Grams, B.: MAIL – eine objektorientierte Sprache für PC, BC und ESER. rechentechnik/datenverarbeitung, Berlin 24 (1987) 6, S. 23

/11/ Weinreb, D.; Moon, D.: Flavors: Message Passing in the LISP-Machine. MIT, AI Laboratory, Memo No. 602, 1980

/12/ Christaller, T.: Ein objektorientierter Ansatz für die Realisierung komplexer Kontrollstrukturen. In: /5/, S. 300

/13/ Fritzsche, H.: Objektorientierte Programmierung auf der Basis von LISP. Tagungsbericht INFO'88, 22.–26. Februar 1988, Dresden, S. 119

/14/ di Primio, F.; Christaller, T.: A Poor Man's Flavor System. Institut Dalle Molle, University Genf, May 1983

/15/ Meyer, B.: Object-Oriented Design. Tutorial Presentation. First European Software Engineering Conference, 8–11 September 1987, Strasbourg, France

/16/ Hendler, J. A.; Wegner, P.: Viewing Object-Oriented Programming As An Enhancement Of Data Abstraction Methodology. Brown University, Dept. of Comp. Sc., Providence, Rhode Island, Technical Report No. CS-85-18, November 1985

/17/ Pascoe, G. A.: Elements of Object-Oriented Programming. Byte 11 (1987) 8, S. 139

/18/ Loeper, H.; Stiller, G.; Jäkel, H.-J.: Betrachtung zur Architektur der Programmiersprachen. Tagungsbericht INFO'88, 22.–26. Februar 1988, Dresden, S. 129

✉ KONTAKT ☏

Technische Universität Dresden, Informatik-Zentrum, Wissenschaftsbereich Systemsoftware, Mommsenstraße 13, Dresden, 8027; Tel. 4 57 54 65

dBase IV

Unter dem Druck starker Konkurrenz anderer Datenbanksysteme will Ashton-Tate seine Führung auf dem Datenbankmarkt mit dem neuen System dBase IV behaupten. Zusätzlich zu den hauptsächlichen neuen Möglichkeiten, wie der Unterstützung von SQL (**S**tructured **Q**uery **L**anguage : Strukturierte Rückfragesprache), einem Quellcode-Compiler und einer wesentlich verbesserten Bedienoberfläche, schlägt dBase IV seine Konkurrenten mit der vollen Kompatibilität zu dBase III. Alle in dBase III erzeugten Datenbanken, Formulare, Reports und Anwendungen laufen ohne Änderung in dBase IV. Die neue Software ist ein wichtiger erster Schritt für die Entwicklung der Verbindung von Datenbanken auf dem PC am Arbeitsplatz und Mainframe-SQL-Datenbanken. Aber, obwohl dBase IV einen kompletten Satz SQL-Befehle entweder im interaktive Kommando-Modus oder innerhalb des dBase-Befehlscodes unterstützt, gibt es keine Möglichkeit des Datentransfers zwischen dBase IV- und SQL-Datenbanken. Also ist die SQL-Implementierung nur für die Entwickler nützlich, die Code schreiben wollen, den sie eventuell auch in SQL-Umgebungen übertragen können.

Echte Übertragbarkeit zwischen SQL- und dBase IV-Datenbanken wird, wie von Microsoft im Januar 1988 angekündigt, mit dem SQL-Server-System für OS/2 zu erwarten sein. Schließlich beabsichtigt Ashton-Tate, dBase zur wichtigsten Datenbanksoftware sowohl auf UNIX- und VMS-Fileservern als auch auf MS-DOS- *und* OS/2-Systemen zu machen.

Wie dBase IV ein Schritt in die Zukunft sein könnte, ist es ebenso ein wichtiger Bruch mit der Vergangenheit. Bis jetzt hatte dBase stets den Ruf, schwer erlernbar und nutzbar zu sein und wesentlich mehr Programmierung als seine Konkurrenten zu benötigen. Man hat diese Kritik mit einer völlig überarbeiteten menügesteuerten Bedienoberfläche beantwortet. Trotzdem kann der vertraute Punkt-Prompt benutzt werden. Es gibt einen verbesserten Anwendungs-Generator, Tastenmakros und eine Möglichkeit der Nachfrage durch Beispiel, vergleichbar mit QBE (**Q**uery-**B**y-**E**xample).

Zusätzlich erhalten die neuen Report- und Formular-Generatoren menügesteuerte Formulargestaltung, berechnete Felder, Datenbewertung, das Zeichnen von Kästen und Linien und andere Möglichkeiten zur Bereitstellung von Ein- und Ausgabeformularen.

Das neue dBase erhält auch eine Option für die automatische Generierung von Grafik auf Ashton-Tates Chart-Master-Grafiksoftware. Man muß dBase verlassen, um Chart-Master zu starten.

Die dBase IV-Datenbankstruktur ist vollkompatibel zu früheren dBase-Versionen, jedoch enthält sie einen neuen Gleitkomma-Datentyp mit einer numerischen Genauigkeit von 64 Bit.

Eine andere Verbesserung des Systems ist die Möglichkeit, verschiedene Datenbanken in einer virtuellen *Pseudo*- oder Betrachtungsdatenbank zu kombinieren, die wie jede andere Datenbank aussieht. Das bedeutet, daß man mit Formularen, Reports und Anwenderprogrammen Zugriff zur Auswertung und Bearbeitung der so verbundenen Datenbanken hat. So werden relationale Operationen wesentlich komplexer als in früheren Versionen. Man kann nur eine Datenbank zur Betrachtung auf dem Bildschirm haben, ohne auf Programmierung zurückgreifen zu müssen.

Das neue System hat auch automatische Indizierung mit bis zu 52 Indizes pro Datenbank in eine Indexdatei. Obwohl man auch noch seine eigenen .NDX-Dateien erzeugen kann, ist dies in den meisten Fällen nicht notwendig.

Das dBase IV-Paket enthält die dBase-Programmier- und Kommandosprache. Diese hat eine Bibliothek von wirtschaftlichen und mathematischen Funktionen, eine *Schablonensprache* zur Nutzung des Anwender-Code-Generators, Kommandos zur Erzeugung von Fenstern mit Pop-up- un Pull-down-Menüs und einem neuen *Mehrfach-Fenster-Debugger.* Die Fenster-Möglichkeit erlaubt dem Systementwickler, Fenster auf dem Bildschirm zu verwenden die mehrere Datenbanken gleichzeitig zeigen.

dBase IV enthält auch einen Compiler. Er erzeugt einen Objektcode, der nach Herstellerangaben dBase-Programme bis zu neunmal schneller ablaufen läßt als in dBase III. Der Compiler legt Dateien mit einer .DBO-Dateierweiterung gegenüber den .COM- oder .EXE-Dateien der meisten anderen Compiler an. Sie sind von dBase IV-Programmen aus startbar. Ein spezielles Werkzeug in der Entwicklerversion gestattet es, die compilierten Programme zusammenzulinken.

Schließlich enthält dBase IV Verbeserungen für Mehrfachnutzer-Anwendungen. Diese ermöglichen Übertragungsprozesse, Fehlerbeseitigung, Schlüsselwort-Datenschutz sowie automatisches *record*- und *file-locking* mit Erkennung der zugriffsberechtigten Arbeitsstation.

dBase IV steht für MS-DOS zur Verfügung, eine OS/2-Version befindet sich in Vorbereitung. Die MS-DOS-Variante erfordert 640 KByte RAM und erkennt und benutzt vorhandene Speichererweiterungen automatisch.

Es gibt verschiedene Versionen für allgemeine Anwendungen und für Systementwickler. Sie sind identisch; nur enthalten die Entwicklerversionen zusätzlich Dokumentationen für Programmierer und Entwickler, das Werkzeug zum Linken des compilierten Codes mit Laufzeitunterstützung sowie LAN-Schlüssel zum Testen von Mehrfachnutzer-Anwendungen.

Ashton-Tate gibt einen mehr als 50 Seiten langen Bericht über die Unterschiede von dBase III und dBase IV heraus, von denen hier nur die wichtigsten zu nennen waren.

Matthias Fischer

TERMINE

X. Fachtagung Mikroelekronik Dresden 1989

Wann? 18. April 1989
Wo? Plenarsaal des neuen Rathauses
WAS?
- Speicherprogrammierbare Steuerungen
- Anwendungsspezifische Schaltkreise
- Mikroelektronik in der Biomedizinischen Technik
- Einsatzbeispiele kundenspezifischer Steuerungen im Werkzeugmaschinenbau
- Lokale Rechnernetze im Industrieeinsatz
- Logistik als überprüfendes Konzept für volkswirtschaftliche Reproduktionsprozesse
- Farbgrafiksystem für CAD-Anwendungen
- Frequenzgesteuerter Elektroantrieb

WIE? Anfragen richten Sie bitte an: VEB Elektroprojekt Dresden, Koll. Böhme, Leningrader Straße 30, PSF 129/130, Dresden, 8012; Tel. 4 86 90

Prof. Dr. Habiger

Zweite Z 1013-Tagung

WER? Computer-Club des VEB Robotron-Anlagenbau Leipzig
WANN? 19. Mai 1989
WO? Leipzig
WIE? Teilnahmewünsche vom 1. März bis 31. März 1989 an Computer-Club robotron, PF 180, Leipzig, 7010

Bachmann

8. Plenarveranstaltung der AG Mikrorechentechnik

WER? Humboldt-Universität zu Berlin
WANN? 3. März 1989
WAS?
- CP/M mit den ZX-Spectrum
- Vorstellung des „BAICODE"-Systems

WIE? Teilnahmewünsche bitte an Direktion für Computerliteratur und Software, Postfach 114, Berlin, 1026.

Steglich

Mikroprozessorsystem K 1810 WM 86

Hardware · Software · Applikation (Teil 6)

Prof. Dr. Bernd-Georg Münzer
(wissenschaftliche Leitung),
Dr. Günter Jorke, Eckhard Engemann,
Wolfgang Kabatzke, Frank Kamrad,
Hellfried Schumacher, Tomasz Stachowiak
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik,
Wissenschaftsbereich Mikrorechentechnik/
Schaltungstechnik

Die Adreßleitungen A16 ... A19 werden mit S3 ... S6-Statussignalen zeitgemultiplext:

S6	S5	S4	S3	Bedeutung
1	1	0	0	DMA on channel 1
1	1	0	1	DMA on channel 2
1	1	1	0	non-DMA on channel 1
1	1	1	1	non-DMA on channel 2

7.6.2 Interne Architektur des IOP
Im Bild 7.9 ist die interne Architektur des I/O-Prozessors dargestellt.

CCU *Common Control Unit*
Alle IOP-Operationen (Befehle, DMA-Transfers usw.) setzen sich aus einer Reihe von internen Buszyklen zusammen, die durch die CCU koordiniert werden. Die CCU kontrolliert auch das Initialisieren des IOP.

ALU *Arithmetic-Logic Unit*
Die ALU kann arithmetische Operationen (Addition, Decrement, Increment) mit 8- oder 16-Bit breiten, nicht vorzeichenbehafteten, binären Zahlen ausführen, wobei das Ergebnis eine binäre Zahl mit maximal 20 Bit sein kann. Zum Befehlsvorrat der ALU gehören auch die logischen Operationen **AND, OR, NOT.**

Assembly/Disassembly Registers
Alle transferierten Daten werden über diese Register geführt. Im Falle eines Datentransfers zwischen zwei Bussen unterschiedlicher Breite fungieren diese Register zur Busanpassung, um zeitlich optimale Datenübertragung zu erreichen.

Beispiel
Quelle: 8 Bit breiter Bus
Ziel: 16 Bit breiter Bus

Für die Quelle werden zwei Lesezyklen durchgeführt, die gelesenen 2 Bytes werden zum 16 Bit Wort zusammengefügt und in einem Schreibzyklus zum Ziel übertragen.

IFU *Instruction Fetch Unit*
Diese Einheit kontrolliert mit Hilfe eines 1-Byte-Queue das Befehlsholen für das aktuelle Kanalprogramm. Bei einem 8-Bit breiten Systembus werden die Befehle byteweise geholt. Für den 16-Bit breiten Bus gibt es zwei Varianten, die in den Bildern 7.10 und 7.11 dargestellt sind.

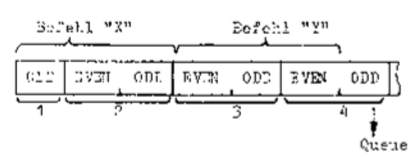


Bild 7.10 Befehlsholen angefangen bei EVEN-Adresse

Fetch Befehlsbyte
- ***1*** ***Die ersten zwei Bytes von „X"***
- ***2*** ***Das dritte Byte von „X" und das erste Byte von „Y", das im Queue zwischengespeichert wird***
- ***3*** ***Das erste Byte von „Y" aus dem Queue: kein Buszyklus***
- ***4*** ***Die letzten 2 Bytes von „Y"***

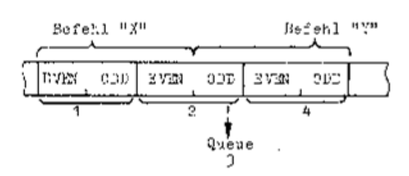


Bild 7.11 Befehlsholen angefangen bei ODD-Adresse

Fetch Befehlsbyte
- ***1*** ***Das erste (ODD-adressierte) Byte (8-Bit-Zyklus) von „X"***
- ***2*** ***Das zweite und dritte Byte von „X"***
- ***3*** ***Das erste und zweite Byte von „Y"***
- ***4*** ***Das dritte Byte von „Y" und das erste Byte vom nächsten Befehl, zwischengespeichert im Queue***

BIU *Bus Interface Unit*
Die BIU führt die Buszyklen aus, steuert den Transfer von Befehlen und Daten zwischen IOP und Speicher bzw. Peripherie.

Channels
Der IOP besitzt zwei unabhängige Kanäle, die sowohl ein eigenes Kanalprogramm, als auch einen DMA-Transfer ausführen können. Jeder Kanal besitzt eine eigene I/O-Steuereinheit, welche die Steuerung während des DMA-Transfers übernimmt. Mit dem DRQ-Signal (DMA-Request) kann der DMA-Transfer synchronisiert werden. Der Kanal wartet auf dieses Signal, bevor er den nächsten Schreib-/Lesezyklus durchführt. Mit dem EXT-Signal kann der laufende DMA-Transfer unterbrochen werden.

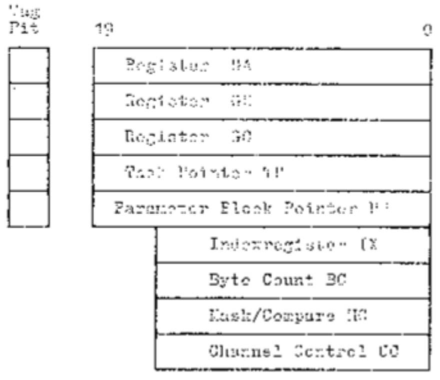


Bild 7.12 Registersatz des 8089

Kanalregisterplatz (Bild 7.12)
GA,GB: allgemeine Register im Kanalprogramm; Ziel oder Quelle im DMA-Transfer
GC: allgemeines Register im Kanalprogramm; zeigt im DMA-Transfer auf das erste Byte der Translationstabelle. Die bei der Quelle gelesenen Daten werden als nicht vorzeichenbehaftete Ziffern interpretiert, die zum Inhalt des GC-Registers addiert werden. Damit entsteht eine Adresse, die auf eines der 256 Bytes der Translationstabelle zeigt. Das Byte wird aus der Translationstabelle entnommen und zum Ziel übertragen (Bild 7.13).
PP: kann nur als Parameter-Block-Pointer genutzt werden und wird während der Initialisierung geladen.
TP: arbeitet als Befehlszähler; sollte im Kanalprogramm nicht verwendet werden.
IX: als allgemeines Register oder als Indexregister verwendbar.
BC: allgemeines Register oder Bytezähler im DMA-Betrieb.

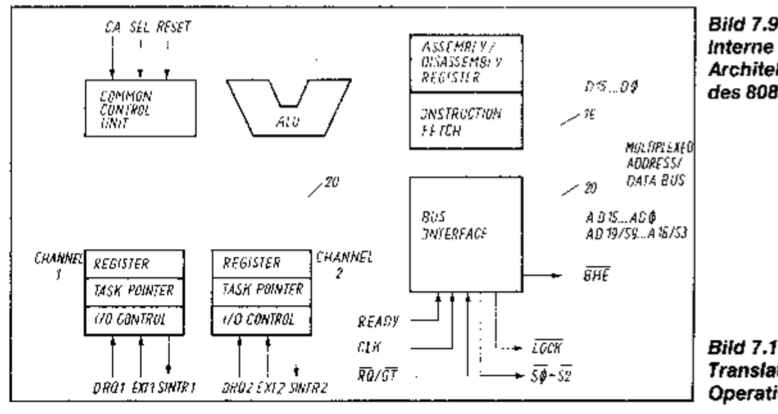


Bild 7.9 Interne Architektur des 8089

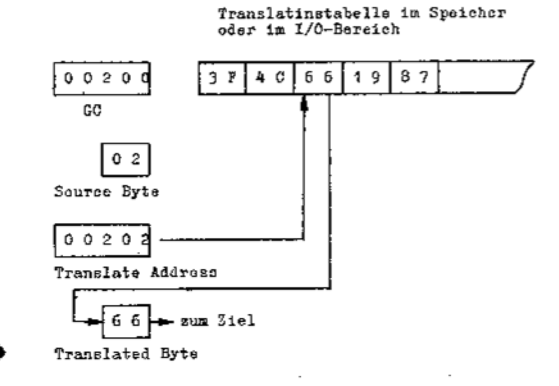


Bild 7.13 Translate Operation

MC: allgemeines Register, auch zum maskierbaren Vergleich verwendbar. Das Low-Byte wird mit dem Vergleichsbyte und das High-Byte mit der Maske geladen. Low in einer Bitposition bedeutet, daß das entsprechende Bit im Vergleichsbyte nicht berücksichtigt wird.

CC: Channel Control (Tafel 7.7)

Dieses Register wird im Kanalprogramm geladen. Der Inhalt des Registers ist in 10 Felder aufgeteilt, die die Informationen über den DMA-Transfer beinhalten.

Tafel 7.7 Channel Control Register

15						7			0
F	TR	SYN	S	L	C	TS	TX	TBC	TMC

F: *Funktion*
- *0 0 Port-zu-Port DMA*
- *0 1 Speicher-zu-Port-DMA*
- *1 0 Port-zu-Speicher-DMA*
- *1 1 Speicher-zu-Speicher-DMA*

TR: *Translation*
- *0 DMA ohne dynamische Übersetzung*
- *1 mit Übersetzung*

SYN: *Synchronisation*
- *0 0 ohne Synchronisation*
- *0 1 Synchronisation durch Quelle*
- *1 0 Synchronisation durch Ziel*
- *1 1 reserviert*

S: *Quelle:*
- *0 GA zeigt auf die Quelle*
- *1 GB zeigt auf die Quelle*

L: *Lock*
- *0 kein Lock während des DMA*
- *1 aktualisiere Lock während DMA*

C: *Chain*
- *0 ohne Chain*
- *1 Chained Programm*

TS: *Abbruch nach einem DMA-Zyklus*
- *0 kein Abbruch*
- *1 Abbruch nach einem Zyklus*

TX: *Abbruch durch externes Signal*
- *0 0 kein Abbruch*
- *0 1 Abbruch Offset – 0*
- *1 0 Abbruch Offset – 4*
- *1 1 Abbruch Offset – 8*

TBC: *Abbruch wenn BC = 0*
- *0 0 kein Abbruch*
- *0 1 Abbruch Offset = 0*
- *1 0 Abbruch Offset = 4*
- *1 1 Abbruch Offset – 8*

TMC: *Abbruch wenn ein Matchbyte gefunden*
- *0 0 0 kein Abbruch*
- *0 0 1 Abbruch Offset – 0*
- *0 1 0 Abbruch Offset – 4*
- *0 1 1 Abbruch Offset – 8*
- *1 0 0 nichts*

Abbruch wenn kein Matchbyte gefunden
- *1 0 1 Abbruch Offset – 0*
- *1 1 0 Abbruch Offset – 4*
- *1 1 1 Abbruch Offset – 8*

TAG-Bit: (Bild 7.12)

Die Register GA, GB, GC und TP sind Pointer-Register, die Zeigeradressen für I/O- oder Speicherbereich beinhalten. Zu jedem der Register gehört ein TAG-Bit, welches zwischen den Bereichen unterscheidet:

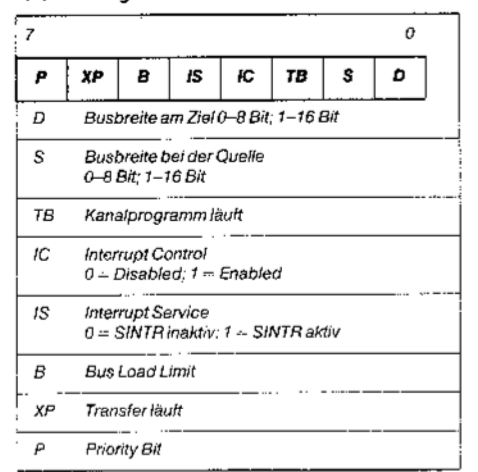
TAG-Bit = 0 ⇒ Speicherbereich

TAG-Bit = 1 ⇒ I/O-Bereich

PSW *Program Status Word* (Tafel 7.8)

Jeder Kanal besitzt ein eigenes PSW, in dem der Zustand des Kanals abgespeichert wird. Dadurch ist es möglich, die Kanalprogramme zu suspendieren und dann durch Regenerierung des alten PSW neu zu starten. Das Kanalprogramm kann auf das PSW nicht zugreifen.

Tafel 7.8 Program Status Word



7							0
P	XP	B	IS	IC	TB	S	D

D	*Busbreite am Ziel 0–8 Bit; 1–16 Bit*
S	*Busbreite bei der Quelle 0–8 Bit; 1–16 Bit*
TB	*Kanalprogramm läuft*
IC	*Interrupt Control 0 – Disabled; 1 – Enabled*
IS	*Interrupt Service 0 = SINTR inaktiv; 1 – SINTR aktiv*
B	*Bus Load Limit*
XP	*Transfer läuft*
P	*Priority Bit*

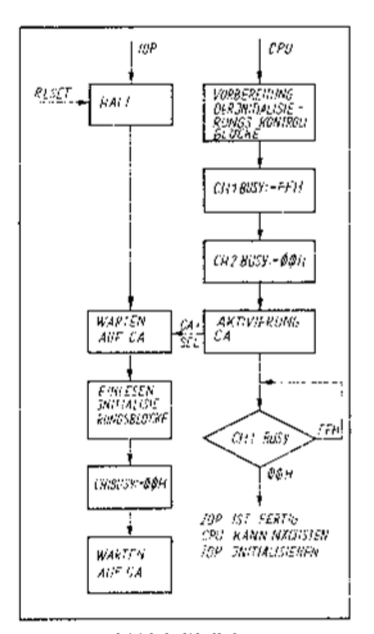


Bild 7.14 8089-Initialisierungssequenz

7.6.3 *Initialisierung des IOP*

Bevor die IOP-Kanäle die Steuerung der I/O-Geräte übernehmen, müssen sie initialisiert werden (Bild 7.14). Der IOP wird zum Einlesen seiner Initialisierungsblöcke durch die CA- und SEL-Signale (CA = Channel Attention, SEL = Auswahl des Kanals) aktiviert (Bild 7.1). Während der Initialisierung erfolgt die Master-Slave-Zuordnung:

- SEL = 0; Kanal 1; IOP programmiert als Master
- SEL = 1; Kanal 2; IOP programmiert als Slave

Die Kanäle des IOP können im I/O-Bereich oder im Speicherbereich der CPU plaziert werden. Im ersten Fall erfolgt das Aktivieren mit einem OUT- und im zweiten Fall mit einem MOV-Befehl.

In der Initialisierungssequenz liest der IOP die sich im Speicher befindenden Initialisierungsblöcke ein (Bild 7.15). Der Inhalt der residenten Anweisungen wird beim Programmieren der EPROMs festgelegt, der RAM-Teil des Initialisierungssteuerblockes muß vor dem IOP-Start durch ein CPU-Programm geladen werden.

- *System Configuration Pointer (SCP),* der mit dem SYSBUS-Byte die Systembusbreite festlegt (Bild 7.16) und mit SCB Segment Base und Offset den Anfang des System-Configuration-Blockes (SCB) angibt.
- *System Configuration Block (SCB)* In diesem Block befindet sich das SOC-Byte (Bild 7.17), mit dem die I/O-Busbreite und Request/Grant-Mode festgelegt wird. Mit dem CB Segment Base und Offset wird die Anfangsadresse des Channel-Control-Blockes angegeben.
- *Channel Control Block (CB)* In diesem Block wird jedem Kanal sein Channel Control Word CCW (Tafel 7.9) und die Fixierung des BUSY-Flagbytes mitgeteilt. Weiterhin bekommt jeder Kanal mit dem PB Segment Base und Offset die Adresse seines Parameter-Blockes.

– *BUSY* ist ein Flagbyte, wird vor der Initialisierung von der CPU mit 0FFH geladen und

Bild 7.15 Initialisierungssteuerblöcke

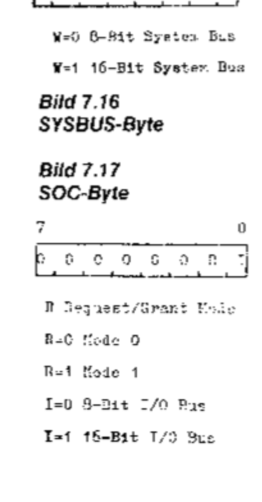
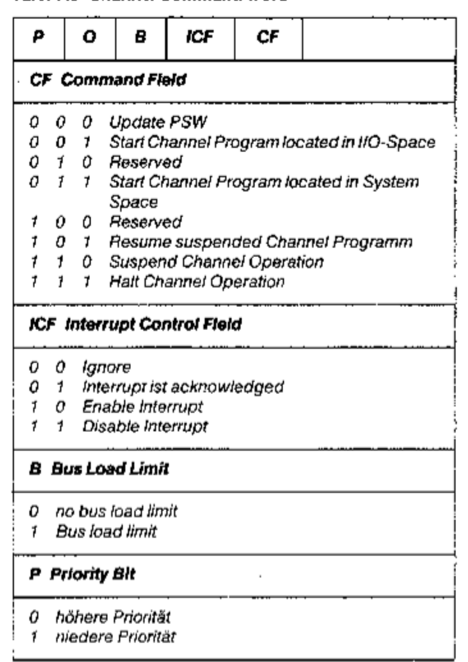


Bild 7.16 SYSBUS-Byte

Bild 7.17 SOC-Byte

Tafel 7.9 **Channel Command Word**



P	O	B	ICF	CF

CF Command Field

0	0	0	Update PSW
0	0	1	Start Channel Program located in I/O-Space
0	1	0	Reserved
0	1	1	Start Channel Program located in System Space
1	0	0	Reserved
1	0	1	Resume suspended Channel Programm
1	1	0	Suspend Channel Operation
1	1	1	Halt Channel Operation

ICF Interrupt Control Field

0	0	Ignore
0	1	Interrupt ist acknowledged
1	0	Enable Interrupt
1	1	Disable Interrupt

B Bus Load Limit

0	no bus load limit
1	Bus load limit

P Priority Bit

0	höhere Priorität
1	niedere Priorität

nach der Initialisierung vom IOP mit 00H überschrieben. Im Kanalprogramm wird das Byte vom IOP 0FFH gesetzt und am Ende mit 00H zurückgesetzt. Mit diesem Flag wird die Aufgabenzuweisung von der 8086-CPU gesteuert.

– *CCW Channel Command Word:* Nach der Initialisierung wird mit jeder CA-Aktivierung das im CCW kodierte Kommando für Kanal 1 oder 2 gelesen und ausgeführt. Mit dem P-Bit wird die Priorität des Kanals festgelegt. (Tafel 7.9)

– *PB: (Parameter Block)* beinhaltet die Startadresse des Kanalprogramms und dient auch zur Zwischenspeicherung von Kanalzustand (PSW) und Task Pointer (TP) mit seinem TAG-Bit, welches das Suspendieren und Neustarten von Kanalprogrammen ermöglicht.

7.6.4 Kanalprogramm

Nach der Initialisierungsphase kommt die Ausführungsphase, die mit einem OUT- oder MOV-Befehl gestartet wird. Die Kanalzuweisung erfolgt mit SEL (SEL = 0 Kanal 1; SEL = 1 Kanal 2). Nach Erkennen des CA-Signals führt der adressierte Kanal folgende Aktivitäten durch:

– BUSY-Flag setzen
– Einlesen des CCW vom Control Block (CB)
– Start und Ausführen der im CCW festgelegten Operation
(Bilder 7.18, 7.19, 7.20, 7.21)
– Rücksetzen des BUSY-Flags nach Programm- oder DMA-Transferende.

7.6.5 DMA-Transfer

Der IOP 8089 realisiert einen DMA-Wort-Datentransfer mit der Geschwindigkeit von 1,25 MByte/s.

Der DMA-Transfer wird vom Kanalprogramm aus mit der Anweisung XFER gestartet. Der danachfolgende Befehl wird noch abgearbeitet, und dann geht der IOP in die DMA-Phase über. Dieser zusätzliche Befehl kann zum Beispiel das letzte Steuerwort in einen Peripheriecontroller laden.

Im DMA-Transfer können eine Reihe von Abbruchbedingungen ausgelöst werden (siehe Channel Control Register und Bild 7.7). Nachdem eine Abbruchbedingung erkannt wurde, wird zum aktuellen Task Pointer der im CC-Register programmierte Offset 0, 4 oder 8 addiert. An diesen berechneten Adressen befinden sich LJUMP-Befehle (Long Unconditional Jump) zu Abbruchserviceprogrammen (Bild 7.22).

7.6.6 Interrupts

Jeder Kanal besitzt die Möglichkeit, über die Leitungen SINTR 1 und 2 eine Interruptanmeldung an die CPU zu senden. Die Interruptsperre bzw. -freigabe erfolgt durch das ICF-Feld des CCW (Tafel 7.9). Nachdem ein Kanal einen Interrupt ausgelöst hat, muß in der Interrupt-Serviceroutine die Interruptanforderung bestätigt und zurückgesetzt werden. Dies erfolgt mit dem Laden eines neuen CCW, in dem das ICF-Feld den logischen Wert „0,1" (interrupt acknowledge) trägt. Nachdem der Kanal das neue CCW empfangen hat, wird das „Interrupt Service Bit" im PSW und die SINTR-Leitung zurückgesetzt.

7.6.7 Konkurrierende Kanaloperationen

Beide Kanäle des IOP können zwar gleichzeitig aktiviert werden, jedoch die aktuelle Buszuweisung übernimmt die CCU in der Prioritätenfolge nach Tafel 7.10. Falls beide Kanäle Operationen mit identischer Priorität

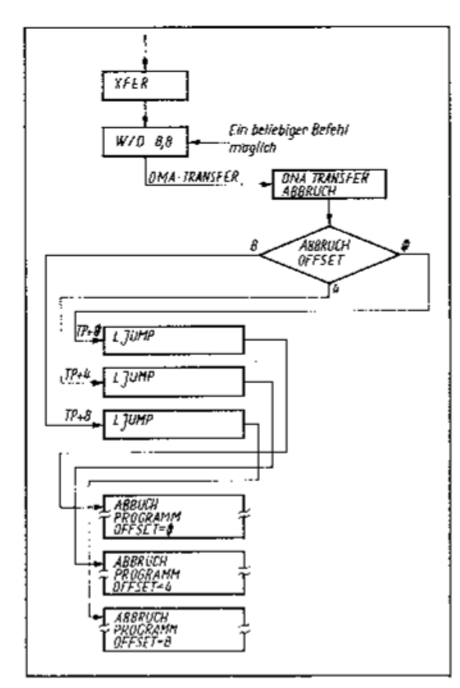


Bild 7.22 Abbruchverhalten des 8089

Tafel 7.10
Zusammenstellung der 8089-Aktivitätsprioritäten

Kanalaktivität	Priorität
DMA-Transfer	1
DMA-Abbruchsequenz	1
Kanalprogramm (Chained)	1
Kanalaktivierung (CA)	2
Kanalprogramm (not chained)	3
untätig (idle)	4

ausführen sollen, erfolgt die Entscheidung an Hand des Prioritätsbits in den PSWs (Tafel 7.8). Wenn diese Prioritätsbits wiederum gleich sind, wird von der CCU ein alternierendes Freigeben der Kanalaktivitäten vorgenommen. Die CCU kann die Kanalaktivitäten nur zu bestimmten Zeitpunkten (interleave boundaries) unterbrechen (Tafel 7.11).

Spezielle Bedingungen

Während LOCK aktiv ist, kann der DMA-Transfer nicht unterbrochen werden.

Kanalprogramm „chained" oder „not chained" kann unterbrochen werden, mit Ausnahme des TSL-Befehls.

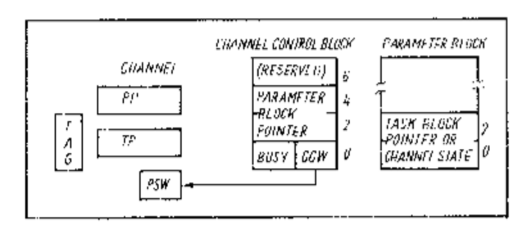


Bild 7.18 Update PSW

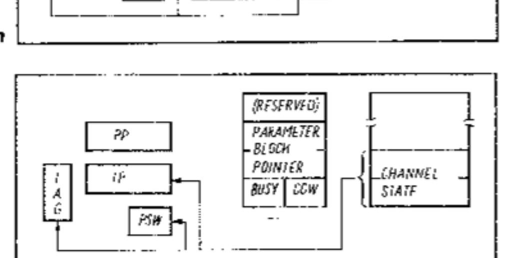


Bild 7.19 Start Program

Bild 7.20 Suspend Operation

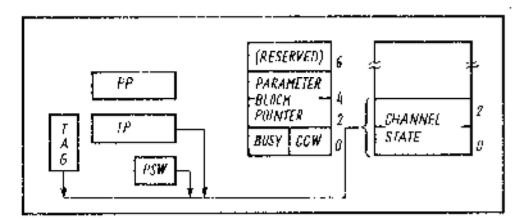


Bild 7.21 Resume Operation

Tafel 7.11 Unterbrechungszeitpunkte (interleave boundaries)

Kanal Aktivität	Priorität	Moment der Unterbrechung durch	
		DMA	Befehl
DMA	1	nach Bus-Zyklus	Bus-Zyklus
DMA-Abbruch Sequenz	1	interner Zyklus	keine
Kanal-Programm (chained)	1	interner Zyklus	Befehl
Channel Attention Sequenz	2	interner Zyklus	keine
Kanal Programm (not chained)	3	interner Zyklus	Befehl
untätig	4	zwei Takte	zwei Takte

7.6.8 *Multiprocessing-Eigenschaften*

Arbitrierungsprinzip ist mit dem des Arithmetik-Koprozessors 8087 vergleichbar und wird durch die $\overline{RQ}/\overline{GT}$-Signale realisiert. Die $\overline{RQ}/\overline{GT}$-Logik des IOP arbeitet in zwei verschiedenen Betriebsarten, die mit dem SOC-Byte (Bild 7.17) in der Initialisierung festgelegt werden:

Mode 0

$\overline{RQ}/\overline{GT}$-Mode 0 ist mit der $\overline{RQ}/\overline{GT}$-Logik der 8086/88-Prozessoren kompatibel (Bild 7.2). Wenn der IOP mit der 8086-CPU im Localmode arbeitet, ist er der Slave und die CPU der Master. In dem Fall, wenn zwei IOPs allein ein Multiprozessorsystem bilden, muß ein IOP während der Initialisierung als Master und der andere als Slave programmiert werden (vgl. Kapitel 7.6.3). Im Mode 0 hat der Masterprozessor keine Möglichkeit, den Bus vom Slave zurückzuverlangen.

Mode 1

Diese Betriebsart dient zum Arbitrieren der Zugriffe auf einen privaten (lokalen) I/O-Bus bei zwei parallel arbeitenden IOPs. In diesem Fall ist ebenfalls ein IOP als Master und der andere als Slave zu programmieren.
Die $\overline{RQ}/\overline{GT}$-Sequenz im Mode 1 lautet:

- Der anfordernde Prozessor sendet einen Request-Impuls.
- Der angeforderte Prozessor gibt den Bus mit einem Grant-Impuls ab.
- Wenn er die Buskontrolle wieder erhalten will, sendet er einen Request-Impuls zwei Takte nach dem Grant-Impuls.

Im Falle eines Multirechnersystems ist die Zugriffsarbitrierung auf gemeinsame Ressourcen mit dem Bus-Arbiter-Schaltkreis 8289 vorzunehmen.

7.6.9 *Assemblerbefehle des 8089*

- Datentransferbefehle

Datentransferbefehle dienen zum Datenaustausch (byte- oder wortweise) zwischen internen Kanalregistern und dem Speicher. Spezielle MOV-Befehle ermöglichen das Laden oder Abspeichern von Adressen und Tagbits in die Zeigerregister GA, GB, GC oder TP (Bild 7.12).

① **MOV** *destination, source*
Entsprechend Datenformat werden vier Gruppen unterschieden:

MOV	Move Word Variable
MOVB	Move Byte Variable
MOVI	Move Word Immediate
MOVBI	Move Byte Immediate

Da diese Befehle das Tagbit des entsprechenden Zeigerregisters setzen, eignen sie sich vor allem für das Laden der I/O-Adressen.

② **MOVP** *destination, source*
(move pointer)
Der MOVP-Befehl transferiert eine physische Adresse und den Wert des Tagbits zwischen Zeigerregistern und Speicher (Bild 7.23).

Zeiger Register

0	2	6	5	F	3	HEX
	0010	0110	0101	1111	0011	Binaer

Speicher

100		101		102		Adresse
F	3	6	5	2	0	HEX
1111	0011	0110	0101	0010	0000	Binaer

Bild 7.23 MOVP-Befehl
○ ***Dieses Feld wird je nach Wert des Tag-Bit mit 0H (Tag-Bit = 0) oder mit 8H (Tag-Bit = 1) überschrieben.***

③ **LPD** *destination, source (load pointer with doubleword)*
Der LPD-Befehl konvertiert einen sich im Speicher befindenden Doppelwortzeiger in eine 20 Bit breite physische Adresse und lädt sie in ein Zeigerregister (Bild 7.24).

100		101		102		103		
6	5	0	0	4	C	3	B	HEX
0110	0101	0000	0000	0100	1100	0011	1011	Binaer

Bild 7.24 LPD-Befehl
Wert des Doppelwortzeigers plaziert unter Adresse 100H:
Segment Base 3B4CH Offset 0065H
Im Zeigerregister geladene physische Adresse lautet: 3B525H

Das Tagbit des Zielregisters wird automatisch zurückgesetzt. Es werden zwei Typen vom LPD-Befehl unterschieden:

LPD	Load Pointer With Doubleword Variable
LPDI	Load Pointer With Doubleword Immediate

- Arithmetische Befehle

Die arithmetischen Befehle interpretieren alle Operanden als nicht vorzeichenbehaftete binäre Zahlen der Breite 8, 16 oder 20 Bit.
Der 8089 besitzt folgende Befehlsgruppen:
① **ADD** *destination, source*
Es werden folgende Additionsbefehle unterschieden, wobei das Ergebnis der Addition im Zieloperanden steht:

ADD	Add Word Variable
ADDB	Add Byte Variable
ADDI	Add Word Immediate
ADDBI	Add Byte Immediate

② **INC** *destination*
Inkrement des Ziels um 1.

INC	Increment Word
INCB	Increment Byte

③ **DEC** *destination*
Dekrement des Ziels um 1.

DEC	Decrement Word
DECB	Decrement Byte

- Logische und Bitmanipulationsbefehle

Die logischen Befehle des 8089 lauten: AND, OR, NOT, wobei die 4 höchsten Bits eines 20-Bit-Zielregisters undefiniert bleiben. Wenn ein Register das Ziel einer Byteoperation ist, werden die höherwertigen Bits 8–15 mit dem Wert des Bits 7 aufgefüllt.
① **AND** *destination, source*
OR *destination, source*
Zwei Operanden werden AND/OR-verknüpft, und das Ergebnis steht im Ziel.

AND/OR	Logical AND/OR Word Variable
ANDB/ORB	Logical AND/OR Byte Variable
ANDI/ORI	Logical AND/OR Word Immediate
ANDBI/ORBI	Logical AND/OR Byte Immediate

② **NOT** *destination, destination/source*
Der NOT-Befehl invertiert die einzelnen Bits des Operanden. Im Falle eines Operanden wird dieser vom Ergebnis überschrieben. Im Fall von zwei Operanden wird die negierte Quelle ins Ziel geschrieben (Ziel muß ein Register sein), der Ausgangswert bleibt erhalten. Der Befehl arbeitet sowohl mit Wort- als auch mit Byte-Operanden:

NOT	Logical NOT Word
NOTB	Logical NOT Byte

③ **SETB** *destination, bit-select*
Dieser Befehl wird zum Setzen von einzelnen Bits im Speicher benutzt. Der Bit-Select-Operand wählt das zu setzende Bit aus.
④ **CLR** *destination, bit-select*
Dieser Befehl führt ein Rücksetzen des ausgewählten Bits durch.

- Sprung und Aufrufbefehle

Das Register TP (Task Pointer) kontrolliert das sequentielle Abarbeiten der Befehle ähnlich einem Befehlszähler. Bei Programmverzweigungen wird zum TP die vorzeichenbehaftete Verschiebung (displacement) addiert. Diese Verschiebung kann eine 8-Bit- (short) oder 16-Bit-Zahl (long, Mnemonic L) sein. Das höchste Bit der Verschiebung ist das Vorzeichen (0 ⇒ positive, 1 ⇒ negative Verschiebung).
– **CALL/LCALL TPsave, target**
Dieser Befehl speichert den aktuellen TP-

Wert und das entsprechende Tagbit im TP-save-Operanden ab. Die Adresse der CALL-Routine wird durch Addition des aktuellen TP und des Target-Operanden berechnet.
Es wird empfohlen, den Rücksprung ins Hauptprogramm mit dem MOVP-Befehl zu realisieren, um so den alten geretteten TP-Wert wieder zu laden.
– **JMP/LJMP** *target*
Mit dem JMP-Befehl wird ein unbedingter Sprung ausgeführt, wobei der TP-Wert nicht gerettet wird.
– **JZ/LJZ** *source, target*
Der Sprung wird ausgeführt, wenn der Source-Operand Null ist, anderenfalls geht das Kanalprogramm zum nächsten Befehl über.
Wenn der Source-Operand ein 20-bit-Register ist, werden nur die 16 niederwertigen Bits ausgewertet.
– **JNZ/LJNZ** *source, target*
Der Sprung wird ausgeführt, wenn der Source-Operand ungleich Null ist.
– **JMCE/LJMCE** *source, target*
Dieser Befehl realisiert einen maskierbaren Vergleich zwischen dem Source-Byteoperanden und dem Inhalt des Mask-Compare-Registers. Das niederwertige Byte des MC-Registers ist das Vergleichsbyte und das höherwertige die Maske. Falls der Vergleich positiv ist, wird ein Sprung realisiert.
– **JMCNE/LJMCNE** *source, target*
Der Sprung wird ausgeführt, wenn das Vergleichsergebnis negativ ist.
– **JBT/LJBT** *source, bit-select, target*
Der Befehl testet das durch „Bit-Select" bestimmte Bit im Source-Operanden. Der Sprung wird ausgeführt, wenn das Bit gleich 1 ist.
– **JNBT/LJNBT** *source, bit-select, target*
Der Sprung wird ausgeführt, falls das selektierte Bit gleich 0 ist.
• Steuerbefehle
Die Steuerbefehle realisieren für das Kanalprogramm die Steuerung der Ausgänge LOCK und SINTR 1–2, das Initialisieren des DMA-Transfers oder das Festlegen der logischen Busbreite.
Beispiele:
– **TSL** *destination, set-value, target*
Mit diesem Befehl kann der Zugriff auf geteilte Ressourcen in einem Multiprozessorsystem gesteuert werden (Bild 7.25). Der Befehl aktiviert das LOCK-Signal, das zum Beispiel mit dem Busarbiter 8289 verbunden werden kann und veranlaßt den Busarbiter, während aktivem LOCK den Bus keinem anderen Master abzugeben. Der TSL-Befehl testet und setzt mit „set-value" die mit „destination" adressierte Speicherzelle. Der Befehl kann zur Implementierung einer Semaphorvariable dienen, mit der der Zugriff auf geteilte Ressourcen eines Multiprozessorsystems gesteuert wird. Wenn der „target"-Operand die Adresse des TSL-Befehls selbst ist, befindet sich das Kanalprogramm so lange in der TSL-Test-Schleife, bis „destination" den Wert 00H besitzt.
– **WID** *source-width, dest-width*
Der Befehl verändert die Bits 0 und 1 des PSW und legt somit die logische Busbreite der Quelle und des Zieles bei einem DMA-Transfer fest.

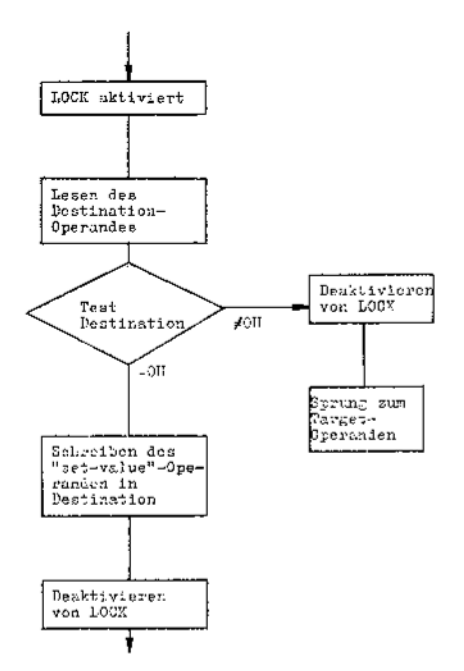


Bild 7.25 TSL-Befehl

Nach dem RESET sind die logischen Busbreiten undefiniert, so daß vor dem ersten Transfer der WID-Befehl ausgeführt werden muß.
– **XFER**
Mit diesem Befehl wird der DMA-Transfer nach dem dem XFER folgenden Befehl gestartet.
– **SINTR**
Dieser Befehl setzt das Interrupt-Service-Bit im PSW und aktiviert die SINTR-Ausgänge, falls das Interrupt-Control-Bit des PSW gesetzt ist.
– **NOP**
Leerbefehl
– **HLT**
Die Arbeit des Kanalprogramms wird angehalten, und das Kanal-BUSY-Byte wird zurückgesetzt.
• Beispielprogramm

LPDI GA, 1E00:0000H	; Anfangsadresse des Quellblockes für DMA-Transfer
LPDI GB, 1E00:8000H	; Anfangsadresse des Zielblockes für DMA-Transfer
MOVI BC, 0AH	; Anzahl der zu transferierenden Bytes
MOVI CC, 0C208H	; Laden des Channel-Control-Registers
XFER	; Start DMA
WID 16,16	; Quelle: 16 Bit breit Ziel: 16 Bit breit
HLT	

Das Beispielprogramm realisiert die Anfangsinitialisierung des IOP für einen Datentransfer von 10 (0AH) Bytes zwischen zwei mit den Registern GA und GB festgelegten Speicherblöcken. Die Startadresse des Programms lautet:
2000:100H
Das CC-Register wird mit dem Wert C208H geladen, was im Detail folgende Informationen für den IOP trägt:
– kein maskierbarer Vergleich
– Abbruch bei BC = 0; Offset 0
– keine Wirkung vom EXT-Signal
– kein Abbruch nach einem Transferzyklus
– „no chaining"
– LOCK aktiv während des Transfers
– GA adressiert die Quelle
– keine Synchronisation (DRQ unwirksam)
– keine Translate-Operation
– Speicher-Speicher-Transfer
Damit das Kanalprogramm ausgeführt werden kann, müssen die Initialisierungssteuerblöcke im Speicher entsprechend geladen werden (Bild 7.15).

8. Überblick zum System 80286

Dieser Abschnitt gibt einen Ausblick auf das 16-Bit-Mikroprozessorsystem 80286 als eine Weiterentwicklung des Systems 8086 /1/, /2/. Im Mittelpunkt stehen dabei die neuen Eigenschaften der CPU 80286, die dem Prozessor im Vergleich zum 8086 eine wesentlich höhere Leistungsfähigkeit verleihen:
• komplexere CPU-Architektur
• integrierte Speicherverwaltungseinheit
• physischer Adreßraum von 16 MByte, virtueller Adreßraum von 1 GByte je Task
• mehrere Speicherschutzfunktionen
• Unterstützung der Taskverwaltung
Diese Leistungsmerkmale sind besonders auf die Belange von Multiuser- und Multitask-Systemen ausgerichtet.

8.1 CPU-Architektur

Die CPU 80286 besteht aus vier Funktionseinheiten (Bild 8.1).
Die **Bus-Unit** (BU) stellt die Schnittstelle zum externen 80286-Bus her. Der 16-bit-Datenbus und der 24-Bit-Adreßbus sind getrennt herausgeführt, wodurch auch ein zeitliches Überlappen der Buszyklen möglich wird. Die Adresse für den nächsten Buszyklus wird bereits ausgegeben, bevor der aktuelle Buszyklus beendet ist. Dieses sogenannte Pipelined-Address-Timing am CPU-Bus des 80286 kann ausgenutzt werden, um bei einer hohen Busbandbreite auch relativ große Speicherzugriffszeiten zuzulassen.
Die BU des 80286 organisiert, genau wie die des 8086, ein vorausschauendes Befehlsholen. Die Befehlsbytes werden in eine 6 Byte tiefe Warteschlange (Prefetch Queue) eingespeist.
Zwischen BU und **Execution Unit** (EU) liegt noch eine zusätzliche Funktionseinheit, die **Instruction Unit** (IU). Diese dekodiert die von der BU kommenden Befehlsbytes und verwaltet eine weitere Warteschlange (Instruction-Queue), in der maximal drei dekodierte Befehle vor ihrer Abarbeitung in der EU zwischengespeichert werden.
Die **Address-Unit** (AU) beinhaltet eine leistungsfähige Speicherverwaltungseinheit, deren Funktionen im Abschnitt 8.3. näher erläutert werden.
Alle vier Funktionseinheiten arbeiten parallel zueinander, wodurch ein hoher Datendurchsatz erreicht wird.
Der Registersatz der 80286-CPU (Bilder 8.2 bis 8.4) enthält neben den von der 8086-CPU bekannten Hauptregistern, Base- und Indexregistern, dem Instruction-Pointer und dem Flagregister eine Reihe zusätzlicher Register, die im folgenden beschrieben werden.

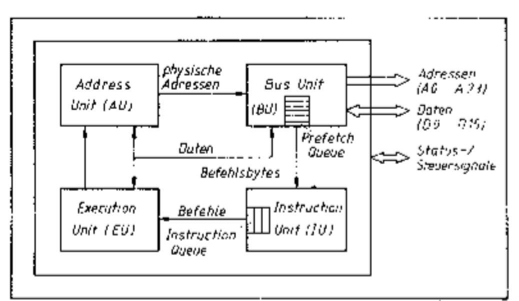


Bild 8.1 Blockschaltbild der CPU 80286

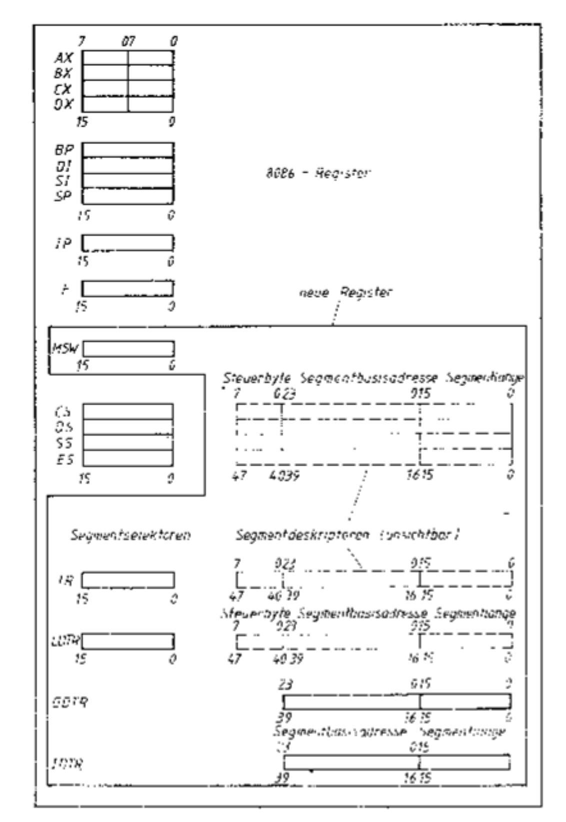


Bild 8.2 Registersatz der CPU 80286

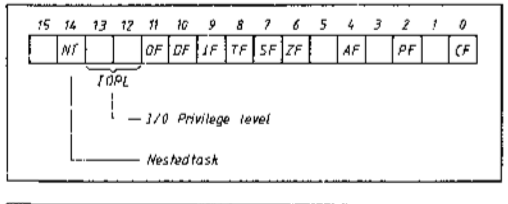


Bild 8.3 Flagregister der CPU 80286

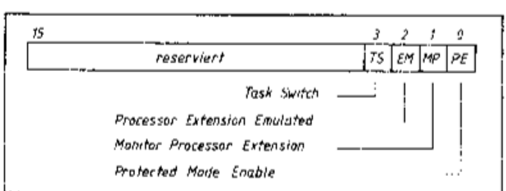


Bild 8.4 Maschinenstatuswort der CPU 80286

8.2 Betriebsarten

Die CPU 80286 kann in zwei Betriebsarten arbeiten: **Real-Address-Mode** (8086-Mode) und **Protected-Mode** (Protected-Virtual-Address-Mode).

Der Befehlssatz des 80286 ist voll abwärtskompatibel zu dem des 8086. Er enthält sämtliche 8086-Befehle und etwa 25 Ergänzungen. Einige dieser zusätzlichen Befehle (z. B. Multiplikationen mit Direktoperand, Block-Ein-/Ausgabe, ...) sind bereits im Real-Address-Mode verwendbar, andere dienen der Speicher- bzw. Taskverwaltung im Protected-Mode.

Nach RESET arbeitet der Prozessor im Real-Address-Mode und wirkt dabei genau wie eine 8086-CPU, das heißt, er unterstützt einen Speicherbereich von 1 MByte. Ein Unterschied besteht jedoch in der Arbeitsgeschwindigkeit. Erstens kann der 80286 mit einer höheren Taktfrequenz betrieben werden (je nach Typ maximal 6 bis 20 MHz), und zweitens erreicht er wegen des verbesserten Pipelinings bei gleicher Taktfrequenz einen etwa um den Faktor 2,5 höheren Datendurchsatz.

Das bedeutet, 8086-Maschinenprogramme sind auf einem 80286-Rechner im Real-Address-Mode ohne Änderungen lauffähig, abgesehen von Zeitbedingungen und der Hardwareumgebung.

Die volle Leistungsfähigkeit erreicht die 80286-CPU im Protected-Mode. Die Umschaltung in diese Betriebsart erfolgt durch das Setzen des Bit *Protected-Mode-Enable* im Maschinenstatuswort (Bild 8.4). Vorher sind jedoch einige Initialisierungen (z. B. Segmentdeskriptor-Tabellen) erforderlich. Eine Rückkehr aus dem Protected-Mode in den Real-Address-Mode ist nur durch RESET möglich.

Im Protected-Mode unterstützt die 80286-CPU einen physischen Adreßraum von 16 MByte (2^{24} Byte) und einen virtuellen Adreßraum von 1 GByte (2^{30} Byte) je Task. Der E/A-Adreßraum umfaßt 64 KByte.

8.3 Speicherverwaltung im Protected-Mode

Die 80286-CPU besitzt eine interne Speicherverwaltungseinheit, welche die Aufgabe hat, die im Programm verwendeten virtuellen Adressen auf den physischen Adreßbereich abzubilden. Gleichzeitig werden eine Reihe von Speicherschutzfunktionen realisiert.

Auch im Protected-Mode wird das Prinzip der Segmentierung verwendet, das heißt, der Adreßraum des 80286 ist in Segmente aufgeteilt, deren Länge von 1 Byte bis 64 KByte variabel ist.

8.3.1 Segment-Deskriptoren

Eine 24-Bit-Speicheradresse wird im Protected-Mode durch lineare Addition von einer 24-Bit-Segment-Basisadresse und einem 16-Bit-Offset gebildet. Der wesentliche Unterschied zum Real-Address-Mode liegt in der neuen Bedeutung der Segmentregister CS, DS, SS und ES. Diese Register enthalten nicht mehr die Segment-Basisadresse, sondern einen Segmentselektor. Deshalb heißen sie im Protected-Mode auch Selektorregister. Wie Bild 8.5 zeigt, dienen Bit 15...3 eines solchen Segmentselektors als Segmentdeskriptor-Index. Das ist ein Index für den dazugehörigen Segmentdeskriptor, der sich innerhalb einer im Speicher angelegten Segmentdeskriptor-Tabelle befindet. Dieser Segmentdeskriptor enthält nun die 24 Bit breite Segment-Basisadresse, aus der durch Addition mit dem 16-Bit-Offset die physische Speicheradresse (A0...A24) gebildet wird.

Bild 8.6 veranschaulicht diesen Vorgang.

Der Umweg über die Segmentdeskriptor-Tabelle erwirkt einige Vorteile. Bild 8.7 zeigt den Aufbau eines Segmentdeskriptors. Er hat eine Länge von 8 Byte und enthält neben der Segment-Basisadresse noch die Länge des Segments und ein Steuerbyte, das den Typ des Segments und die Zugriffsrechte definiert. In den Bildern 8.8 und 8.9 sind die Formate der Steuerbytes für Daten- und Codesegmente angegeben. Die Angaben im Steuerbyte ermöglichen mehrere Speicherschutzfunktionen. Bei jedem Speicherzugriff wird automatisch die Einhaltung der festgelegten Segmentgrenzen (Länge) und Zugriffsrechte überprüft. Verletzungen dieser Bedingungen lösen spezielle Interrupts (Exceptions) aus.

Um die Speicherzugriffe durch die beschriebenen Vorgänge nicht zu verlangsamen, werden die aktuellen Segmentdeskriptoren im Prozessor zwischengespeichert. Im Protected-Mode bestehen die vier Segmentregister aus den Segmentselektor-Registern (CS, DS, SS, ES) und zusätzlich je einem unsichtbaren, 48 Bit breiten Segmentdeskriptor-(Cache-)Register (Bild 8.2). Dieses enthält eine Kopie des vom jeweiligen Segmentselektor ausgewählten Segmentdeskriptors.

Die Segmentselektoren werden durch die vom 8086 bekannten Befehle wie LDS, POP ES, JMPF (intersegment) beeinflußt. Im Protected-Mode lösen diese Befehle automatisch das Kopieren des neuen Segmentdeskriptors vom Speicher in das unsichtbare CPU-Cache-Register aus. Damit stehen der internen Speicherverwaltungseinheit während der folgenden Zugriffe auf dieses Segment die notwendigen Angaben zur Verfügung.

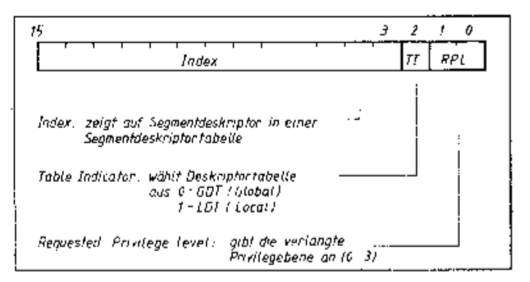


Bild 8.5 *Aufbau eines Segmentselektors*

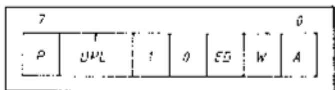


Bild 8.7 *Segmentdeskriptor*

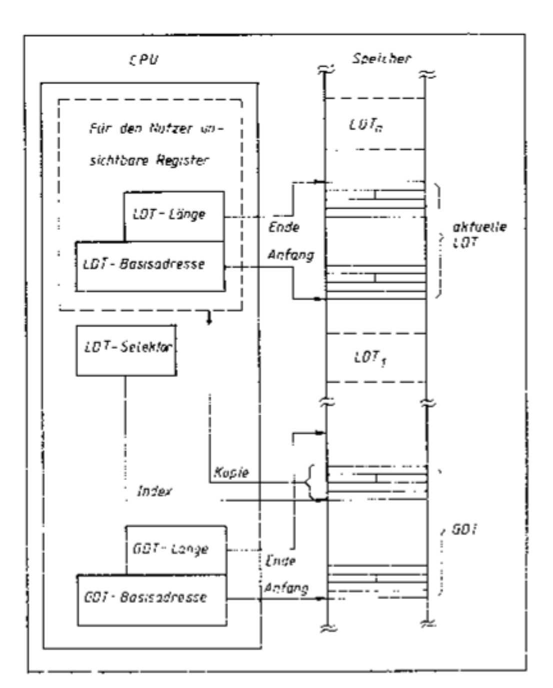


Bild 8.10 *Globale und lokale Deskriptortabellen*

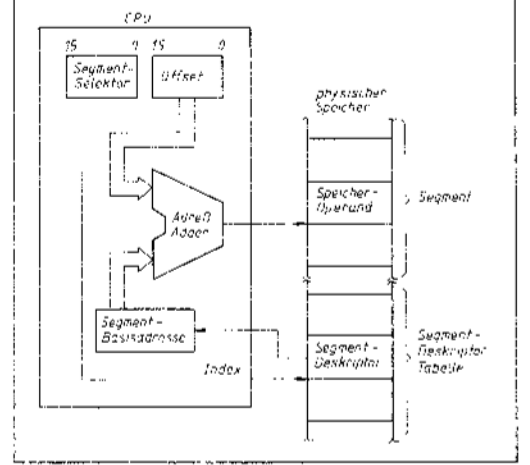


Bild 8.6
Speicheradressierung im Protected-Mode

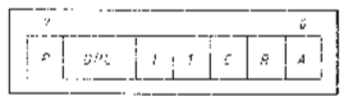


Bild 8.8 Steuerbyte
eines Datensegment-Deskriptors
Bit 7: P (Present) 1-Segment ist im Speicher vorhanden
Bit 6,5: DPL (Descriptor Privilege Level)
Privilegstufe 0 ... 3
Bit 4 = ED (Expansion Direction)
0-Ausdehnung nach oben
Offset ≤ Limit
1-Ausdehnung nach unten
Offset > Limit (z. B. Stack)
Bit 1: W (Writeable) 0 – Read Only
1 – Read/Write
Bit 0: A (Accessed) 1 – Segmentdeskriptor wurde bereits benutzt

P | DPL | 1 | 1 | C | R | A

Bild 8.9 Steuerbyte
eines Codesegment-Deskriptors
Bit 4 = 1, Bit 3 = 1:
Kennzeichen für Codesegment
Bit 2: C (Conforming) 1-Segment paßt sich der aufrufenden Privilegebene an
Bit 1: R (Readable)
0-Segment ist nur ausführbar
1-Segment ist ausführbar und lesbar

8.3.2 Deskriptortabellen

• Globale und lokale Deskriptortabellen

80286-Programme werden in Programmteile, sogenannte Tasks, mit getrennten Speicherzugriffsrechten aufgeteilt.

Die Segment-Deskriptoren aller Speichersegmente, die von einer Task benutzt werden, sind in zwei Deskriptortabellen enthalten (Bild 8.10). Die **Global-Deskriptor-Table** (GDT) enthält die Deskriptoren der von allen Tasks nutzbaren globalen Segmente. Die **Local-Descriptor-Table** (LDT) enthält Deskriptoren von privaten Segmenten einer Task. Die Auswahl zwischen beiden Deskriptortabellen wird durch das TI-Bit im Segmentselektor getroffen (vergleiche Bild 8.5).

Mit den 13 Bit des Segmentdeskriptor-Index werden 2^{13} = 8192 Segmentdeskriptoren (je 8 Byte) in einer Segmentdeskriptor-Tabelle adressiert. Somit stehen einer Task maximal 2^{14} Speichersegmente (8192 globale und 8192 lokale) mit einer Länge von je 64 KByte (2^{16} Byte) zur Verfügung. Daraus ergibt sich ein virtueller Adreßraum von einem Gigabyte (2^{30} Byte) je Task.

Die Deskriptortabellen sind selbst Speichersegmente. Ihre Lage innerhalb des physischen Speichers wird in entsprechenden CPU-Registern festgelegt (vergleiche Bild 8.2). Das **Global-Descriptor-Table-Register** (GDTR) enthält die Segment-Basisadresse (24 Bit) im absoluten Speicherraum und die Länge (16 Bit) der GDT. Das GDTR kann mit dem Load-Befehl LGDT geladen werden.

Das **Local-Descriptor-Table-Register** (LDTR) ist genau wie die oben beschriebenen Segmentregister aufgebaut. In einem sichtbaren 16-Bit-LDT-Selektor vom Format nach Bild 8.5 wird durch den Index die Lage des LDT-Deskriptors in der GDT festgelegt. In einem verdeckten 40-Bit-Cache-Register wird eine Kopie des aktuellen LDT-Deskriptors mitgeführt. Mit dem Befehl LLDT wird nur der Selektor der aktuellen LDT geladen, der mit dem Index zum Deskriptor der aktuellen LDT zeigt. Das Kopieren des entsprechenden Segmentdeskriptors, also Steuerbyte, LDT-Basisadresse und -Länge, in die CPU erfolgt dann automatisch. Wie in Bild 8.7 dargestellt, kann es mehrere LDT geben, deren Deskriptoren sich in der GDT im Speicher befinden.

• Interrupt-Deskriptortabelle

Die **Interrupt-Descriptor-Table** (IDT) ist vergleichbar mit der Interrupt-Tabelle des 8086. Sie enthält sogenannte Gate-Deskriptoren (je 8 Byte) für maximal 256 verschiedene Interrupts.

Basisadresse zur Lage der IDT im Speicherraum und Länge der IDT können mit dem Befehl LIDT in das **Interrupt-Descriptor-Table-Register** (IDTR) geladen werden (Bild 8.2).

8.4 Privilegkonzept

Die CPU 80286 gestattet nicht nur eine einfache Trennung von Systemprogrammen und Anwenderprogrammen, sondern unterstützt vier Software-Privilegebenen. Diese sind von 0 bis 3 numeriert, dabei ist Ebene 0 am meisten privilegiert und Ebene 3 am wenigsten (Bild 8.11).

Vom Privilegkonzept des 80286 sollen hier nur die wichtigsten Prinzipien erläutert werden.

Jedem Segment wird eine der vier Privilegebenen zugeordnet und als Descriptor-Privilege-Level (DPL) im Steuerbyte des Segmentdeskriptors kodiert (Bilder 8.8 und 8.9).

Die Privilegebene des aktuell ausgeführten Programms, **Current-Privilege-Level** (CPL), wird im RPL-Feld (vergleiche Bild 8.5) des aktuellen Codesegment-Selektors vorgegeben, also in Bit 1 und 0 des CS-Registers. Innerhalb einer Task muß CPL also nicht konstant bleiben, sondern kann sich, abhängig vom gerade ausgeführten Programm (Codesegment), ändern.

Für Datenzugriffe gilt nun die Regel, daß Programme nur auf Datensegmente der gleichen oder einer niederen Privilegebene zugreifen können. Es muß also gelten:

CPL des aktuellen Codesegments ≤ DPL des Datensegment-Deskriptors

Ein numerisch kleinerer Wert entspricht einer höheren Ebene.

Bei Codezugriffen muß normalerweise die Regel

CPL des aktuellen Codesegments = DPL des Ziel-Codesegment-Deskriptors

eingehalten werden, das heißt, innerhalb einer Task können mit JMPF (intersegment) oder CALLF (intersegment) nur Codesegmente der gleichen Privilegebene aufgerufen

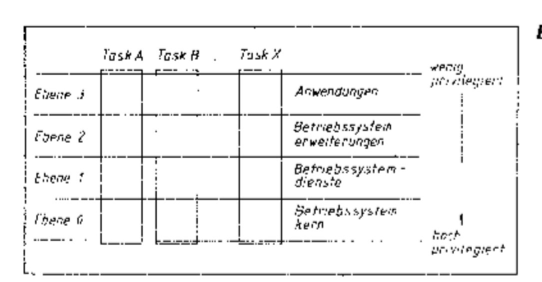


Bild 8.11 Privilegebenen

werden. Allerdings sind folgende zwei Ausnahmen gestattet:

- Codesegmente mit der Eigenschaft *conforming*, zum Beispiel Module des Betriebssystems, können auch von weniger privilegierten Programmen, z. B. Anwenderprogrammen, aufgerufen werden (vergleiche Bild 8.8). Sie passen sich dabei der aufrufenden Privilegebene an.
- Für den kontrollierten Übergang zwischen den Privilegebenen gibt es spezielle Deskriptoren, sogenannte CALL-Gates. Ein solcher Deskriptor (ähnlich wie in Bild 8.7) enthält den Selektor des Ziel-Codesegments (also den Verweis auf den entsprechenden Codesegment-Deskriptor) und den Offset des Eintrittspunktes in dieses Codesegment. Für den Aufruf eines CALL-Gate gelten die Bedingungen:

CPL des aktuellen Codesegments ≦ DPL des CALL-Gate-Deskriptors

CPL des aktuellen Codesegments ≦ DPL des Ziel-Codesegment-Deskriptors

Zur Durchsetzung des Privilegkonzeptes gibt es weiterhin einige privilegierte Befehle, die nur von Programmen der Privilegebene 0 ausgeführt werden dürfen. Dazu gehören zum Beispiel die Ladebefehle für die CPU-Register IDTR, GDTR und LDTR.
Ein- und Ausgabebefehle sind ebenfalls privilegiert. Die niedrigste Privilegebene, in der diese Befehle noch ausgeführt werden dürfen, wird als **I/O-Privilege-Level** (IOPL) bezeichnet und kann durch Bit 13 und 12 des Flagregisters festgelegt werden (Bild 8.3). Es muß also gelten:

CPL des aktuellen Codesegments ≦ IOPL

Die Einhaltung aller genannten Regeln wird von der CPU automatisch überprüft. Verletzungen lösen spezielle Status-Interrupts (Exceptions) aus.

8.5 Taskverwaltung

Ein weiteres wichtiges Leistungsmerkmal der 80286-CPU im Protected-Mode ist die interne Taskverwaltung.
Diese darf jedoch nicht als ein auf dem Chip installiertes Multitask-Betriebssystem verstanden werden. Die CPU erledigt nur die bei jedem Taskwechsel notwendigen Grundoperationen, das heißt, das Retten des alten und Laden des neuen CPU-Inhaltes. Die eigentliche Taskverwaltung, also die Entscheidung, wann welcher Taskwechsel erfolgt, muß nach wie vor softwaremäßig vorgenommen werden.

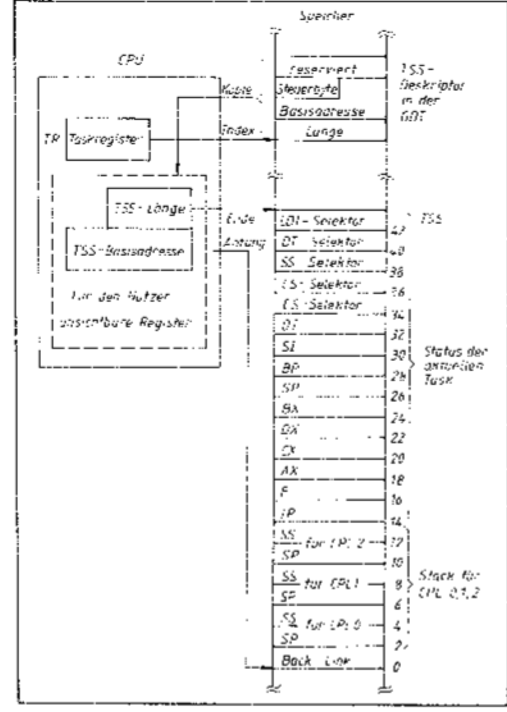


Bild 8.12
Task-State-Segment und TSS-Deskriptor

8.5.1 Task-State-Segment

Zu jeder Task gehört ein **Task-State-Segment** (TSS; Bild 8.12). Das ist ein Speicherbereich, in dem der Abarbeitungsstand der jeweiligen Task festgehalten wird. Dazu gehören Registerinhalte, Adreßbereiche (Segmentselektoren) und ein Verweis auf die vorhergehende Task (Back-Link). Die aktuelle Task wird durch ein spezielles Selektor-Register, das Task-Register TR, ausgewählt (vergleiche Bild 8.2). Dieses Register enthält einen Selektor, der nach dem bekannten Prinzip auf einen TSS-Deskriptor innerhalb der GDT verweist. Der Deskriptor enthält Basisadresse, Länge und Steuerbyte des aktuellen TSS. Diese Informationen werden auch hier in einem verdeckten CPU-Register mitgeführt.

8.5.2. Taskumschaltung

Eine Taskumschaltung erfolgt nun durch den Ansprung des entsprechenden Task-State-Segment mit einem Befehl JMPF (intersegment) oder CALLF (intersegment) oder über ein Task-Gate. Daraufhin werden automatisch (vergleiche Bild 8.12)

- alle Register in das aktuelle TSS gerettet
- das Task-Register mit dem neuen Selektor geladen
- der TSS-Deskriptor in den unsichtbaren Teil des Task-Registers kopiert
- die CPU-Register mit den Werten aus dem neuen TSS geladen
- die Abarbeitung der neuen Task dort fortgesetzt, wo sie zuletzt unterbrochen wurde.

Bei der Taskumschaltung wird außerdem in das Back-Link-Feld des neuen TSS der Selektor des alten TSS (alter TR-Inhalt) eingetragen. Das ist zum Beispiel dann von Bedeutung, wenn die Taskumschaltung von einem Interrupt ausgelöst wurde. Als Kennzeichen dafür dient das Bit NT (Nested-Task) im Flagregister (Bild 8.3). Bei gesetztem NT-Flag wird der Befehl IRET den Back-Link nutzen, um wieder eine Umschaltung auf die alte, durch den Interrupt unterbrochene Task auszulösen.

8.5.3 Task-Gates

Alle Task-State-Segment-Deskriptoren müssen in der GDT stehen. Damit Tasks auch über die LDT bzw. durch Interrupts auch über die IDT aufgerufen werden können, gibt es sogenannte Task-Gates. Das sind Vermittler-Deskriptoren, ähnlich den CALL-Gates, die außer dem Steuerbyte nur den Selektor des dazugehörigen TSS-Deskriptors enthalten. Task-Gate-Deskriptoren können in der GDT, der LDT und der IDT stehen und dienen dort als Verweis auf einen TSS-Deskriptor und damit auf ein Task-State-Segment.

Literatur

/1/ The 8086 Family User's Manual 1980
/2/ User Manual Intel 1979
/3/ 8087 Numeric Data Coprocessor. Intel Corporation 1986
/4/ Mikrokomputery szesnastobitowe, 6. Szkola Mikroprocesorowa, 3.–5. Dezember 1984. Polskie Towarzystwo Informatyczne Centrum Szkolenia Informatycznego ZETO, Łodz
/5/ Microsystem Components Handbook. Microprocessors Volume I. Intel 1986
/6/ Vieillefond, C.: Programmierung des 80286. Düsseldorf: SYBEX-Verlag 1987

wird fortgesetzt

8 × 8 – ein Font für alle Fälle

Thomas Bauer, Berlin

Um eine Grafik mit Schrift aufzuwerten, bedient man sich heute unterschiedlicher Methoden. Vielfach werden Pixel-Zeichensätze verwendet. Für niedrigauflösende Grafik mit geringen Ansprüchen reicht dabei ein einfacher 4 × 6-Zeichensatz (4 × 6-Font) häufig schon aus. Der hier vorzustellende 8 × 8-Font bietet gegenüber dem 4 × 6-Font bei höher auflösender Grafik eine bessere Konturschärfe bei gleicher Schriftgröße.
Die meisten Programmiersprachen bieten standardmäßig die Möglichkeit, Schriftzeichen in beliebiger Größe und Richtung in der Grafik auf einfache Weise zu plazieren. Dabei werden – z. B. in Turbo-Pascal-Version 4.0 – bisweilen gleich mehrere Fonts zur Verfügung gestellt.

Für einige Anwendungen ist es von Vorteil, wenn die Sache mit dem Zeichensatz für den Programmierer transparent ist. Oft ist es in solch einem Fall unerläßlich, auf einen selbstdefinierten Font zurückzugreifen. Ein Beispiel dafür dürfte die reine Druckergrafik sein, bei der der „Grafikbildschirm" unsichtbar als ein ARRAY im RAM verwaltet wird und mit den Standardroutinen der Programmiersprachen für die Bildschirmgrafik nicht behandelbar ist. Auch um der Option willen, eigene Zeichen (z. B. für die E-Technik) definieren zu können, kann so ein Font der Marke Eigenbau Vorteile bringen.
Der hier vorgestellte 8 × 8-Font (siehe 2. und 3. Umschlagseite), der der Standardschrift auf der CGA-Karte entspricht, kann sowohl für die Bildschirmgrafik als auch für die Druckergrafik Verwendung finden. Die beiden Prozeduren zum Zeichnen des Textes erlauben vertikales und horizontales Schreiben in verschiedener Schriftgröße (siehe Demo auf der 3. Umschlagseite). Der Vergrößerungsfaktor (SIZE) vergrößert die Schrift dabei in Vielfachen von 8 × 8-Pixeln. Das normale 8 × 8-Pixel-Format wird bei SIZE = 1 erreicht.
Die ASCII-Zeichen jenseits von 167 sind vor allem dann von Vorteil, wenn es darum geht, schnell und problemlos dicke Linien und Umrandungen in die Grafik zu bringen. Die Routinen in Bild 1 sowie die FONT-Definition lassen sich analog auch in anderen Programmiersprachen (z. B. Basic, Fortran) leicht formulieren.

Literatur

/1/ Der erweiterte Zeichensatz des PC. DOS, DMV-Verlag (1988) 3, S. 73

/2/ Plate, J.: Hochauflösende Drucker-Grafik. mc, Fanzis-Verlag (1986) 9, S.62

***Bild 1** DEMO-Programm zur Anwendung des 8 × 8-Font auf einer CGA-Karte (Bild 3 siehe 2. und 3. Umschlagseite)*

```
PROGRAM FON_DEMO;

CONST font8x8 : ARRAY[0..255,0..7] OF 0..$FF =
  (($00,$00,$00,$00,$00,$00,$00,$00)  (* #00 *),
   ($7e,$81,$99,$bd,$81,$a5,$81,$7E)  (* #01 *),
   ($7E,$ff,$E7,$C3,$FF,$DB,$FF,$7E)  (* #02 *),
   ($00,$10,$38,$7C,$FE,$FE,$FE,$6C)  (* #03 *),
   ($00,$10,$38,$7C,$FE,$7C,$38,$10)  (* #04 *),
   ($7C,$38,$7C,$FE,$FE,$38,$7C,$38)  (* #05 *),
   ($7C,$38,$7C,$FE,$7C,$38,$10,$10)  (* #06 *),
            ..                  ..

 { Aus Platzgründen kann die Definition des Pixel-
   zeichensatzes hier nur andeutungsweise mitgeteilt
   werden. Alle zur programmtechnischen Realisierung
   notwendigen Informationen können leicht Bild 3
   entnommen werden.                                 }

            ..                  ..
   ($00,$00,$00,$18,$00,$00,$00,$00)  (*  ·  *),
   ($1C,$3C,$6C,$EC,$0C,$0C,$0C,$0F)  (*  √  *),
   ($00,$00,$00,$6C,$6C,$6C,$6C,$78)  (*  ⁿ  *),
   ($00,$00,$00,$78,$60,$30,$18,$70)  (*  ²  *),
   ($00,$00,$3C,$3C,$3C,$3C,$00,$00)  (*  ■  *),
   ($00,$00,$00,$00,$00,$00,$00,$00)  (*  ?  *));

TYPE tempstr   = STRING[80];
     direction = (ltor,upwards,downwards);
     (* direction gibt die Textrichtung an:
           ltor      : left to right - normale Schrift
           upwards   : vertikal aufwärts
           downwards: vertikal abwärts                      *)
VAR i : INTEGER;
    s : STRING[255];

(* ------------------------------------------------------ *)

PROCEDURE xplot(x,y:INTEGER);
(* Hier kann die für die Grafikkarte oder die Druckergrafik
   spezifische PLOT-Prozedur angeschlossen werden.
   Für dieses Beispiel wurde die TURBO-PASCAL-PLOT-Prozedur
   für die CGA-Karte herangezogen.                          *)
BEGIN
  plot(x,199-y,1)
END;

(* ------------------------------------------------------ *)

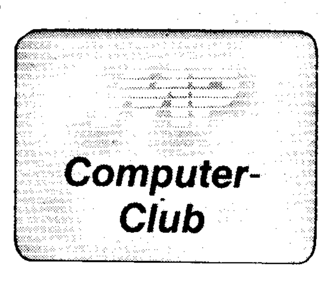
PROCEDURE drawchar(x,y:INTEGER; ch:CHAR; size:BYTE;
                      d:direction);
CONST bitmap : ARRAY[0..7] OF BYTE=(1,2,4,8,16,32,64,128);
VAR   i,j,k,l,c,xs,xe,ys,ye : INTEGER;
BEGIN
  c:=ord(ch);
  IF c IN [0..255] THEN
    FOR i:=0 TO 7 DO FOR j:=0 TO 7 DO
    IF (font8x8[c,i] AND bitmap[7-j])<>0 THEN BEGIN
      CASE d OF
        ltor      : BEGIN
                     xs:=x+j*size; xe:=xs+size-1;
                     ys:=y+i*size; ye:=ys+size-1;
                     FOR l:=ys TO ye DO
                       FOR k:=xs TO xe DO xplot(k,l);
                    END;
        upwards   : BEGIN
                      xs:=y+j*size; xe:=xs+size-1;
                      ys:=x-i*size; ye:=ys-size+1;
                      FOR l:=ys DOWNTO ye DO
                        FOR k:=xs TO xe DO xplot(l,k);
                    END;
        downwards: BEGIN
                      xs:=y-j*size; xe:=xs-size+1;
                      ys:=x+i*size; ye:=ys+size-1;
                      FOR l:=ys TO ye DO
                        FOR k:=xs DOWNTO xe DO xplot(l,k);
                    END;
      END   (*  CASE D of ...                        *)
    END;    (*  IF (font8x8[c,i] and bitmap ...       *)
END;

(* ------------------------------------------------------ *)

PROCEDURE drawstring(x,y:INTEGER; s:tempstr; Size:BYTE;
                        dir:direction);
(* die x-,y-Koordinaten der Textposition beziehen sich
   stets auf die linke untere Ecke des ersten Buchstabens
   des Strings.                                             *)
VAR k : INTEGER;
BEGIN
  CASE dir OF
  ltor      : FOR k:=1 TO length(s) DO
              drawchar(x+(k-1)*8*size,y,s[k],size,ltor);
  upwards   : FOR k:=1 TO length(s) DO
              drawchar(x,y+(k-1)*8*size,s[k],size,upwards);
  downwards: FOR k:=1 TO length(s) DO
              drawchar(x,y-(k-1)*8*size,s[k],size,downwards);
  END
END;

(* -------------------- D E M O -------------------------- *)

BEGIN    (* DEMO für CGA-Karte *)
  clrscr;
  s:='';
  FOR i:=1 TO 255 DO s:=s+chr(i);
  hires;
  drawstring(0,0,'╚═══════════════════╝',4,ltor);
  drawstring(0,160,'╔═══════════════════╗',4,ltor);
  drawstring(0,128,'║',4,ltor);
  drawstring(608,128,'║',4,ltor);
  drawstring(0,96,'╟───────────────┬──╢',4,ltor);
  drawstring(0,64,'║ │',4,ltor);
  drawstring(544,64,'│ ║',4,ltor);
  drawstring(0,32,'║ │',4,ltor);
  drawstring(544,32,'│ ║',4,ltor);
  drawstring(59,124,' M P  - Demo ',5,ltor);
  drawstring(279,116,'(Size = 5)',1,ltor);
  drawstring(0,191,
  ' 8x8 - Pixelzeichensatz                   DEMO',1,LTOR);
  drawstring(47,38,'vertikal',1,upwards);
  drawstring(65,38,'upwards!',1,upwards);
  drawstring(595,103,'vertikal',1,downwards);
  drawstring(578,103,'downwards!',1,downwards);
  drawstring(96,98,
  'Die Ziffern in doppelter Größe (Size=2):',1,ltor);
  drawstring(96,80,copy(s,48,10),2,ltor);
  drawstring(96,70,
 'Der erweiterte Zeichensatz des PC in einfacher Größe:',
  1,ltor);
  drawstring(96,60,copy(s,1,56),1,ltor);
  drawstring(96,52,copy(s,57,56),1,ltor);
  drawstring(96,44,copy(s,113,56),1,ltor);
  drawstring(96,36,copy(s,169,56),1,ltor);
  drawstring(96,28,copy(s,224,56),1,ltor);
  REPEAT UNTIL keypressed;
  textmode
END.
```



Ein Kurs MASCHINEN-CODE im Rundfunk

Seit Januar des Jahres gibt es im Schulfunk von Radio DDR II die neue Sendereihe *Computer-Magazin* (mittwochs, 17.00 Uhr, 14täglich). Die Sendungen haben das Ziel, aktuell über Computertechnik und -literatur zu informieren und Anregungen zum Einsatz von Kleincomputern zu geben. Außerdem werden in bewährter Form Programme zum Mitschneiden angeboten.

Ab März wird innerhalb des Computer-Magazins ein Grundkurs zur Programmierung in Maschinen-Code für den U 880 (Z 80) durchgeführt. Autor und Lektor ist Prof. Dr. Horst Völz. Hier eine kurze Übersicht über die Themen des Kurses (Terminänderungen sind vorbehalten):

Thema	*Sendedatum (17.00 Uhr, DDR II)*
1. Hilfsmittel und Methoden	8. 3. 89
2. Hardware und Befehlstypen	22. 3. 89
3. Die ersten Befehle (Lade-, Austausch-, *Jump*-Befehle usw.)	5. 4. 89
4. Arithmetik, *Shift* und *Rotate*	19. 4. 89
5. Relativsprünge und Bitbefehle	3. 5. 89
6. Weitere Befehle (*In, Out, Restart* usw.)	18. 5. 89
7. Verallgemeinerungen (Debugging, Assembler usw.)	31. 5. 89

Zum besseren Verständnis der Sendungen wird vom Rundfunk ein Begleitmaterial zum gesamten Kurs angeboten. Dieses können Sie erhalten, wenn Sie einen frankierten und an sich selbst adressierten A5-Umschlag (nur dieses Format!) schicken an:
Schulfunk, Radio DDR, Nalepastraße, Berlin, 1160; Kennwort: Maschinen-Code

Meinungen, Wünsche und Angebote zum Computer-Magazin sind natürlich ebenfalls gefragt.

Dr. Joachim Baumann
Schulfunk

SCP-Disketten-Directory als REDABAS-Datei

In bestimmten Anwendungsfällen ist es wünschenswert, das Inhaltsverzeichnis einer Diskette als Datenfile zur Verfügung zu haben. Leider ermöglicht das Betriebssystem SCP die Erzeugung einer derartigen Datei auf Betriebssystemniveau nicht.

Das im folgenden vorgestellte TURBO-PASCAL-Programm erzeugt eine mit REDABAS auswertbare Datei, die das Inhaltsverzeichnis einer unter SCP angelegten Diskette enthält. Da das Programm auf Betriebssystemroutinen des SCP zugreift, läuft es nur unter diesem bzw. kompatiblen Betriebssystemen (nur 8 Bit). Das Programm ist so wie abgedruckt lauffähig. Allerdings sind eine Reihe von Ergänzungen, vor allem zur Fehlerbehandlung oder eine individuelle Anpassung denkbar und sinnvoll. Einige dementsprechende Möglichkeiten wurden im Programm bereits verwirklicht.

```
program directory;
type Pointer = ^Eintr;
     Eintr   = Record
                 Satz   :string[21];
                 Next   :Pointer;
               end;
     Name    = string[14];
var  FirstEintrag,Eintrag,LastEintrag:Pointer;
     FileVar   :text;
     DMA_Puffer:array [1..128] of byte;
     FCB       :array [1..36] of byte;
     k         :integer;
     Diskname  :string[8];
     FileName  :Name;
     Lw        :char;

procedure FillFCB (var FileName:Name);
var n,PunktPos:integer;
begin
  PunktPos:=pos('.',FileName);
  for n:=3 to PunktPos-1 do
    FCB[n-1]:= ord( FileName[n]);
  for n:=PunktPos+1 to PunktPos+3 do
    FCB[n-2]:= ord( FileName[n]);
   FCB[1]:=ord (lw)-64;
   FCB[13]:=0;
end;

function Existiert (var FileName:Name):boolean;
var f:integer;
begin
  FillFCB (FileName);
  Existiert:=bdos(17,addr(FCB)) in [0..3];
end;

procedure Erzeuge (var FileName:Name);
begin
  FillFCB (FileName);
  writeln ('Welchen Namen soll die Diskette erhalten');
  readln (Diskname);
  assign (FileVar,FileName);
  rewrite (FileVar);
  writeln (FileVar,Diskname);
  close (FileVar);
end;

procedure SpeichernEintrag (k:integer);
var n:integer;
begin
  New (Eintrag);
  Eintrag^.satz:= Diskname + ',';
  for n:=1 to 8 do
    Eintrag^.satz:= Eintrag^.satz + chr (DMA_Puffer[32*k+1+n]);
  Eintrag^.satz:= Eintrag^.satz + ',';
  for n:=9 to 11 do
    Eintrag^.satz:= Eintrag^.satz + chr (DMA_Puffer[32*k+1+n]);
  if FirstEintrag = nil then FirstEintrag:=Eintrag
                        else LastEintrag^.Next:=Eintrag;
  LastEintrag:=Eintrag;
  LastEintrag^.Next:=nil;
end;

 procedure Schreiben;
 begin
 assign (FileVar,'DIR.TXT');
 rewrite (FileVar);
 while FirstEintrag <> nil do
 begin
   writeln(FileVar,FirstEintrag^.satz);
   FirstEintrag:=FirstEintrag^.Next;
 end;
 close(FileVar);
 end;

 begin
   bdos($1a,addr(DMA_Puffer));       {Zuweisung neue DMA-Adresse}
   write ('Directory von Laufwerk: ');
   readln (Lw);
   if Lw > 'Z' then Lw:= chr( ord(Lw) - 32);
   FileName:=Lw + ':' + 'DISKNAME.XXX';
   if Existiert (FileName) then
   begin
     assign (FileVar,FileName);  {Einlesen des Diskettennamens}
     reset (FileVar);
     read (FileVar,Diskname);
     close (FileVar);
   end
   else
     Erzeuge (FileName);
   FirstEintrag:=nil;
   FileName:=Lw + ':' + '????????.???';
   FillFCB (FileName);
   k:=bdos($11,addr(FCB));            {Ersten Eintrag suchen}
   while k in [0..3] do
   begin
     SpeichernEintrag(k);             {gefundenen Eintrag abspeichern}
     k:=bdos(18,addr(FCB));           {naechsten Eintrag suchen}
   end;
   Schreiben;  {gefundene Eintraege als Textdatei auf Diskette}
   bdos($1a,$80);        {Zuweisung Standard- DMA-Adresse}
 end.
```

Programmbeschreibung

Zur Diskettenarbeit benutzt das Betriebssystem zwei Pufferspeicher. Das ist einerseits der sogenannte File-Control-Block (FCB), in dem die zum Lesen oder Schreiben einer Datei notwendigen Daten übergeben werden, zum Beispiel der Name der Datei und das entsprechende Laufwerk. Die Übergabe der zu lesenden oder zu schreibenden Daten erfolgt im DMA-Puffer. Für beide Speicherbereiche existieren im Betriebssystem Vorzugsadressen, die durch den Nutzer aber verändert werden können.

Das Programm nutzt 3 der durch das Betriebssystem bereitgestellten BDOS-Funktionen. Diese Funktionen sind:

- Funktion 17: Diese Funktion sucht im Inhaltsverzeichnis der Diskette den ersten Eintrag, der dem vorher in den FCB eingetragenen Dateinamen entspricht. Dieser Name darf Fragezeichen als Symbol für einen beliebigen Buchstaben enthalten. Als Parameter ist der Funktion die Adresse des FCB zu übergeben. Zurückgegeben wird die Nummer des 32-Byte-Blockes, in dem die gesuchten Dateiinformationen im DMA-Puffer zu finden sind. Dieser Wert liegt im Bereich von 0 bis 3. Wenn die gesuchte Datei nicht gefunden wird, erfolgt die Rückgabe von 255 (0FFH).
- Funktion 18: Diese Funktion sucht den nächsten Eintrag entsprechend den Angaben im FCB. Diese Funktion darf nur nach der Funktion 17 ausgeführt werden. Die zu übergebenden und zu empfangenden Parameter entsprechen denen der Funktion 17.
- Funktion 26: Diese Funktion setzt die neue Adresse des DMA-Puffers.

Vom TURBO-PASCAL aus werden die Funktionen mit der Funktion BDOS(i, j) aufgerufen. Dabei ist i die entsprechende Nummer der Funktion und j der zu übergebende Parameter. Näheres zu den BDOS-Funktionen ist entsprechender Literatur zu entnehmen.

Das Programm selbst ist relativ einfach aufgebaut. Unmittelbar zu Beginn wird die entsprechende DMA-Adresse zugewiesen. Nach Abfrage des Laufwerkes, von dem das Inhaltsverzeichnis gelesen werden soll, wird auf dem entsprechenden Laufwerk die Datei mit dem Namen DISKNAME.XXX gesucht und bei Vorhandensein der in ihr enthaltene Diskettenname in die Variable *Diskname* eingelesen. Auf diese Weise ist die softwaremäßige Unterscheidung der einzelnen Disketten ohne Zutun des Nutzers möglich. Sollte diese Datei noch nicht vorhanden sein, so wird sie durch das Pogramm angelegt. Der Nutzer wird in diesem Fall zur Eingabe des gewünschten Diskettennamens aufgefordert. Soll der Name der Diskette später geändert werden, so kann dies mit einem beliebigen Editor (z. B. dem Textprozessor) geschehen. Für das Suchen der Datei wird die BDOS-Funktion 17 genutzt.

Anschließend wird der FCB mit Fragezeichen (Byte 2 bis 12) gefüllt. Dabei entsprechen die Positionen 2 bis 9 dem eigentlichen Dateinamen, die Positionen 10 bis 12 der Extension. Werden die letzten 3 Bytes nicht mit Fragezeichen („???"), sondern zum Beispiel mit „COM" gefüllt, wird nur nach Dateien mit der Extension „COM", also ausführbaren Programmen, gesucht.

Die Position 1 im FCB enthält die Laufwerksangabe. Dabei entspricht eine 0 dem aktuellen Laufwerk, eine 1 dem logischen Laufwerk A, eine 2 dem logischen Laufwerk B usw. Das Betriebssystem ermöglicht die Unterscheidung von insgesamt 16 Laufwerken.

Anschließend wird der erste Eintrag gesucht. In der folgenden Schleife erfolgt das Zwischenspeichern der gefundenen Einträge in einer dynamisch aufgebauten Datei.

Sind alle Einträge gefunden, wird der

Inhalt des Zwischenspeichers als Textdatei auf der Diskette unter dem Namen DIR.TXT abgelegt. In dieser Datei sind die einzelnen Datenfelder (Diskettenname, Dateiname, Extension) durch Kommas voneinander getrennt. Dies ermöglicht es, diese Datei von REDABAS aus mit dem Befehl APPEND FROM DIR.TXT DELIMITED in eine REDABAS-Datei einzulesen. Dazu muß im REDABAS vorher eine Datenbankdatei mit entsprechender Struktur erzeugt werden. Die Felder dieser Datenbankdatei müssen gleich lang oder länger als die entsprechenden Strings im TURBO-PASCAL-Programm sein, da ansonsten Zeichen ohne Fehlermeldung unterdrückt werden. Zum Schluß des Programms erfolgt vorsichtshalber die Zuweisung der standardmäßigen DMA-Adresse.

Erprobt wurde das Programm unter den SCP-Versionen 0.4 und 0.5 sowie CP/A. Es eignet sich besonders zur nachträglichen Erfassung großer Datenbestände innerhalb eines Programms zur Diskettenverwaltung. Einzelne Routinen können in anderen Programmen genutzt werden, zum Beispiel der Test auf Vorhandensein einer Datei.

Bernd Matzke

RAM-Disk für den KC 87

Die im folgenden beschriebene RAM-Diskette gestattet es, daß mit einem KC abwechselnd mit mehreren Programmen gearbeitet werden kann, ohne daß ständig von Kassette nachgeladen werden muß. Dies ist insbesondere dann vorteilhaft, wenn ein oder mehrere Nutzer an mehreren Aufgaben arbeiten.

Bedingungen

① Programmende unterhalb 3FFFH; bei Basic-Programmen ist deshalb die maximale Programmlänge auf 15 Kilobyte begrenzt.

② Maximale Aufrüstung des KC mit 2 RAM-Erweiterungsmodulen.

③ Es muß sich um Basic-Programme handeln, Maschinenprogramme sind bedingt abspeicherbar.

Anweisungen und ihre Funktion

Die folgenden Anweisungen sind im Betriebssystem (OS-Modus) einzugeben. Vom Basic wird das Betriebssystem mit dem Basic-Befehl **BYE** erreicht. Mit der Anweisung **WBASIC** gelangt man zurück zum Basic, ohne vorhandene Basic-Programme zu zerstören. Beim erstmaligen Start des Basic-Interpreters ist der maximale Speicherraum auf 3FFFH zu begrenzen, indem beispielsweise die Frage *MEMORY SIZE?* mit 16383 beantwortet wird.

NEW: Alle Programme werden gelöscht, bei Neustart unerläßlich.

DSAVE: Zunächst ist ein maximal 8stelliger Name einzutragen. Nach Drücken der ENTER-Taste wird das Programm in den Bereich 4000H ... BFFFH an das zuletzt abgespeicherte Programm geschoben. Würde dabei 4000H unterschritten oder würde die 11 Programme fassende Adreßtabelle überlaufen, so erfolgt die Meldung *VOLL*. Abgespeichert wird dann nicht. Vor der Speicherung von Maschinenprogrammen muß in die Zellen 03D7/8H die Endadresse eingetragen werden. Zu beachten ist, daß immer ab 02F0H abgespeichert wird.

DLOAD: Nach Drücken der ENTER-Taste wird das jeweils vorher abgespeicherte Programm angezeigt. So entsteht eine Liste der Namen aller im RAM gespeicherten Programme. Wird die angezeigte Frage *LOAD (J)* mit *J* beantwortet, so wird dieses Programm in den Arbeitsbereich geladen. Danach ist der Basic-Interpreter mit **WBASIC** aufzurufen.

DELETE: Damit wird das zuletzt geladene Programm gelöscht. In Verbindung mit DLOAD und DSAVE läßt sich auch das vorletzte Programm herauslöschen.

Hinweise zum Maschinenprogramm

Das Programm sollte am obersten Ende des Speicherbereiches stehen, beispielsweise ab BB000H oder ab BE00H. Wegen der Nutzung der transienten Kommandos beim Aufruf muß es auf einer integralen 100H-Grenze beginnen. Unterhalb der Marke *ENDE* wird das erste abgespeicherte Programm geladen. Die mit *ENAK* gekennzeichnete Doppelzelle zeigt auf den Beginn des zuletzt abgespeicherten Programmes. Die mit *ZEIG* bezeichnete Zelle zeigt auf die Adreßtabelle, die nach *ZEIG* beginnt. Diese Adreßtabelle enthält die Anfangsadressen aller abgespeicherten Programme. Weitere Hinweise sind im Quelltext enthalten.

Dr. Wolfgang Nestler

```
        PN RAMDISK
        ORG OBBOOH
ENDE:EQU ANFA-1;UNTERHALB BEGINNT DER ABSPEICHERBREICH
ANFA:EQU OBBOOH;BEGINN DES PROGRAMMES
ENAK:EQU ANFA+030H;ANFANGSADR.DES ZUL. GESP.PROGRAMMES
ZEIG:EQU ENAK+2;ZEIGT AUF DAS ZULETZT GESP. PROGRAMM
        JMP     NEWE
DB 'NEW    '
DB      0
        JMP     DSAV
DB      'DSAVE  '
DB      0
        JMP     DLOD
DB      'DLOAD  '
DB      0
        JMP DLET
DB 'DELETE  '
DB  0
BER      0018H
;RAM DISKETTE NEU INITIALISIEREN
NEWE:   LD      DE,ENDE
        LD      (ENAK),DE
        LD      HL,ZEIG
        INC     HL
        INC     HL
        LD      (ZEIG),HL
        LD      (HL),D
        INC     HL
        LD      (HL),E
        RET
;PROGRAMM IN RAMDISKETTE LADEN
DSAV:   CALL    NAME
        LD      DE,02F0H        ;ANFANG ABSPEICHERBEREICH
        LD      HL,(03D7H)      ;BASIC ENDE
        OR      A
        SBC     HL,DE           ;LÄNGE
        PUSH    HL
        PUSH    HL
        POP     BC
        LD      HL,(ENAK)       ;FREIE OBERE GRENZE
        OR      A
        SBC     HL,BC           ;UNTEN
        LD      A,H
        CMP     3FH             ;SPEICHER VOLL
        JPC     VOLL
        POP     BC
        PUSH    HL
        PUSH    HL
        POP     DE
        LD      HL,02F0H
        LDIR
        POP     HL
        DEC     HL
        LD      (ENAK),HL       ;NEUES ENDE
        LD      DE,(ZEIG)       ;FÜLLSTAND
        INC     DE
        INC     DE
        LD      (ZEIG),DE       ;NEUER FÜLLSTAND
        EX      DE,HL           ;)ADRESSE
        LD      (HL),D          ;)FÜR
        INC     HL              ;)02F0 H
        LD      (HL),E
        LD      A,L
        CMP     40H
        JPP     VOLL            ;ADRESSTABELLE VOLL
        POP     DE
        RET
;SPEICHER ODER ADRESSTABELLE VOLL
VOLL:   LD      C,9
        LD      DE,TEXT1
        CALL    5
        RET
TEXT1:  DB       'VOLL'
DB      0
NAME:   LD      C,9
        LD      DE,TEXT2
        CALL    5
        LD      DE,02F0H
        LD      A,8
        LD      (02F0H),A
        LD      A,0
        LD      (02FAH),A
        LD      C,0AH
        CALL    5
        RET
TEXT2:  DB       'NAME '
DB      0
;ANZEIGEN UND LADEN,WENN J(A) TASTE GEDIPT
DLOD:   LD HL,(ZEIG)
        PUSH HL
        LD      BC,ZEIG         ;ZEIGT LETZTES PROGRAMM
        OR      A
        SBC     HL,BC
        LD      B,L
        SRL     B               ;ZAHL DER PROGRAMME
        DEC     B
        POP     HL
        INC     HL
 DL1:   LD      E,(HL)
        DEC     HL
        LD      D,(HL)
        PUSH    DE
        INC     DE
        INC     DE              ;BEGINN DES PROGRAMMNAMEN
        INC     DE
        LD      C,9
        CALL    5
        DEC     HL
        PUSH    DE
        LD      DE,TEXT3
        CALL    5
        LD      C,1
        CALL    5
        CMP     'J'
        JRZ     LOA-#
        POP     DE
        POP     DE
        DJNZ    DL1-#
        POP     DE
        RET
TEXT3:  DB       ' LOAD?(J)'
DB      0AH
DB      0DH
DB      0
 LOA:   POP     DE              ;LADEN DES PROGRAMMES
        POP     DE
        LD      C,(HL)
        DEC     HL
        LD      B,(HL)
        PUSH    BC
        POP     HL
        OR      A
        SBC     HL,DE
        PUSH    HL
        POP     BC
        PUSH    DE
        POP     HL
        LD      DE,02EFH
        LDIR
        RET
DLET:   LD HL,(ZEIG);LÖSCHEN DES ZUL.GELADENEN PROGRAMMS
        DEC HL
        LD E,(HL)
        DEC HL
        LD D,(HL)
        LD (ZEIG),HL
        LD (ENAK),DE
        RET
END
```

REDABAS-Tip

e-Funktion

Das Berechnungsverfahren für die e-Funktion (Bild 3) beruht auf folgender Schreibweise für die Funktionen $y = e^x = (e^1)^a * (e^{0.1})^b * (e^{0.01})^c * (e^{0.001})^d \ldots$ und $y = e^{-x} = 1/e^x$

Wegen der Verarbeitbarkeit im Rechner werden die Funktionswerte

$e^{-1} = 0.3678794$

$e^{-0.1} = 0.9048374$

$e^{-0.01} = 0.9900498$

$e^{-0.001} = 0.9990005$ benutzt.

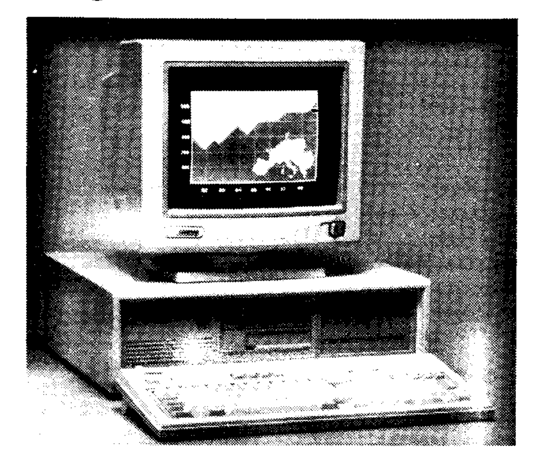
Für kleine Werte von x gilt die Näherung: $y = e^x = 1 + x$

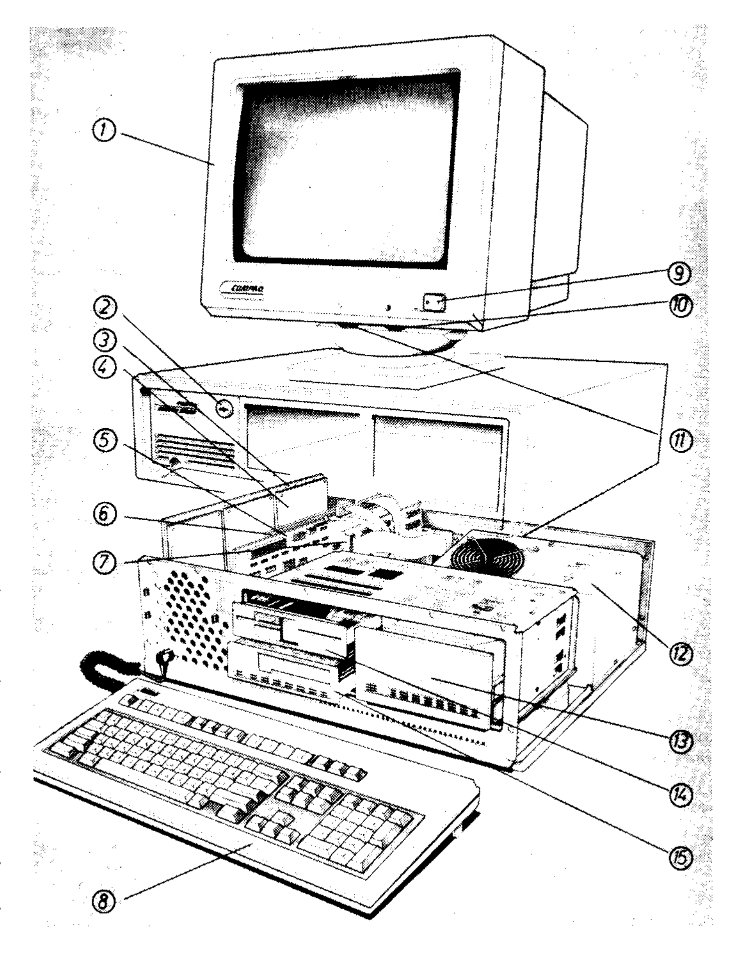
Thomas Steffens

Bild 1 e-Funktion

```
*-------------------------------------------------------------------
* E_FKT.PRG / 11.09.87 / REALISIERUNG DER E-FUNKTION
*-------------------------------------------------------------------
* DAS BERECHNUNGSVERFAHREN IST IN DER BEDIENUNGSANLEITUNG DES
* MINIREX 75,VEB ROEHRENWERK MUEHLHAUSEN ZU FINDEN.
* DER WERTEBEREICH DER EXPONENTEN WIRD NICHT UEBERPRUEFT !
*-------------------------------------------------------------------
* EINGANG: ME1 = < EXPONENT (NUMERISCH,AUCH NEGATIV) >
*
* INTERN : MI1...MI6 = HILFSVARIABLEN
*
* AUSGANG: MA1 = < ERGEBNIS >, MA1 = e hoch ME1
*-------------------------------------------------------------------
IF ME1 < 0
 STORE F TO MI5
 STORE ME1 * -1 TO ME1
ELSE
 STORE T TO MI5
ENDIF
STORE 1 TO MA1
STORE INT(ME1) TO MI1
IF (ME1-MI1) > 0
 STORE STR((ME1-MI1+0.00000000),10,8) TO MI6
 STORE VAL(¤(MI6,3,1)) TO MI2
 STORE VAL(¤(MI6,4,1)) TO MI3
 STORE VAL(¤(MI6,5,1)) TO MI4
 STORE VAL('1.000'+¤(MI6,6,5)) TO MA1
 DO WHILE MI4 > 0
  STORE MA1/0.9990005 TO MA1
  STORE MI4-1 TO MI4
 ENDDO
 DO WHILE MI3 > 0
  STORE MA1/0.9900498 TO MA1
  STORE MI3-1 TO MI3
 ENDDO
 DO WHILE MI2 > 0
  STORE MA1/0.9048374 TO MA1
  STORE MI2-1 TO MI2
 ENDDO
ENDIF
DO WHILE MI1 > 0
 STORE MA1/0.3678794 TO MA1
 STORE MI1-1 TO MI1
ENDDO
IF.NOT.MI5
 STORE 1/MA1 TO MA1
ENDIF
RELEASE MI1,MI2,MI3,MI4,MI5,MI6,ME1
RETURN
*-------------------------------------------------------------------
```

Compaq Deskpro 386/25





1 *Farbmonitor*
2 *Sicherheitsschloß*
3 *Speichergrundplatine*
4 *Speichermodul*
5 *Mehrzweck-Controller*
6 *VG-Steuerkarte*
7 *135-MByte-Bandlaufwerk-Controller*
8 *Enhanced Tastatur*
9 *Netzschalter*
10 *Kontrastregler*
11 *Helligkeitsregler*
12 *Netzteil*
13 *300-MByte-Festplattenlaufwerk*
14 *1,2-MByte-Diskettenlaufwerk*
15 *135-MByte-Bandlaufwerk*

In MP 9/88, S. 287, hatten wir im Zusammenhang mit der Vorstellung neuer IBM-Modelle des PS/2 auf den „Gegenzug" der Firma Compaq – die Ankündigung des 386/25 – hingewiesen. Damit ist IBM nunmehr wiederum, neben anderen Anbietern von 25-MHz-Maschinen, ein ernsthafter Konkurrent im Bereich der Spitzen-PCs erwachsen. Das von drei Firmengründern mit geliehenen 30 Millionen Dollar Startkapital 1982 ins Leben gerufene Unternehmen galt von Anfang an mit der Produktion von MS-DOS-fähigen, billigeren und schnelleren PCs als Widersacher von IBM. Bereits das erste Modell, der tragbare Portable Computer, war so erfolgreich, daß das Startkapital nach einem Jahr zurückgezahlt werden konnte; Rekorde wurden mit dem höchsten Erstjahres- und Zweitjahresergebnis in der amerikanischen Wirtschaftsgeschichte erzielt. Heute beschäftigt das vom Firmenchef Rod Canion geleitete und expandierende Unternehmen weltweit über 4000 Mitarbeiter und ist als Weltmarktführer bei tragbaren PCs anzusehen. Als bemerkenswert und gewagt galt der Entschluß Compaqs, im September 1986 einen PC mit dem neuen 32-Bit-Prozessor von Intel, dem 80386, auf den Markt zu bringen – das heißt, ein halbes Jahr vor der Präsentation von IBMs erstem 32-Bit-PC, dem Modell 80 der völlig neu geschaffenen Linie PS/2.

Das Besondere an den Deskpro-Modellen ist die Beibehaltung der Kompatiblität zu den bisherigen 80286-PCs mit AT-Bus, trotz erheblicher Leistungssteigerung, die das 4fache eines PC AT betragen soll. Basis dafür ist die von Compaq entwickelte Flex-Architektur, ein Mehrwege-Bussystem mit drei unabhängigen und gleichzeitig benutzbaren Datenwegen. Die CPU (Intel 80386/25) kommuniziert mit dem Cache-Controller (Intel 82385) über einen 32 Bit breiten Bus. Dieser liest und schreibt die Daten ebenfalls über einen 32 Bit breiten Bus, wie die CPU mit 25 MHz. Der Datentransfer zu den Massenspeichern und der Peripherie jedoch erfolgt auf dem AT-kompatiblen Bus mit 8 oder 16 Bit Breite, abhängig von der vorhandenen Peripherie. Das heißt, der Steckbus ist vom Bus zwischen CPU und Speicher abgekoppelt, so daß die CPU vom Datentransfer der E/A-Module (mit eigener Intelligenz) unbehelligt bleibt. Diese Architektur machte bereits den Deskpro 386/20 überraschend schnell und erfolgreich. Ein weiteres Merkmal ist die Verwendung eines 2-Wege-Cachespeichers, womit z. B. bei Multitasking der Cache von zwei Tasks gleichzeitig benutzt werden kann. Die zweimal 16 KByte statischen RAMs mit einer Zugriffszeit unter 25 ns erlauben bei mehr als 95 Prozent der Zugriffe eine Arbeit ohne Wartetakte der CPU (null Waitstates).

Zur Leistungserhöhung läßt sich schließlich auch ein mathematischer Koprozessor einsetzen. Dies kann sowohl der 80387 von Intel als auch ein Weitek 3167, beide mit 25 MHz, sein. Während der Weitek 1167 noch aus drei Chips bestand, ist der neue 3167 – auch unter der Bezeichnung Abacus bekannt – eine Einchiplösung und damit platzsparend. Zudem liefert er bei einer Taktfrequenz von 25 MHz und einfacher Genauigkeit eine Leistung von 1 MFLOPS (Millionen Gleitkomma-Operationen pro Sekunde) gegenüber 0,2 MFLOPS des 80387. Die besseren Leistungen des Abacus resultieren aus einer speziellen Architektur, die mittels 64-Bit-Datenpfaden Berechnungen mit einfacher Genauigkeit in 200 ns erlaubt (gegenüber 1 bis 1,5 ms beim 80387). Mit diesen Parametern soll sich ein Deskpro 386/25 mit Abacus bereits in den Bereichen von traditionellen Workstations einsetzen lassen.

Als weitere Einsatzgebiete sind zu nennen:
– CAD/CAE-Anwendungen
– in Netzen als Fileserver
– KI-Systeme
– Softwareentwicklung u. a.

MP-We

Technische Daten

Prozessoren:
– 80386-Mikroprozessor mit 25 MHz Taktrate
– Cache-Controller 82385 mit 32 KByte statischem RAM (25ns)
– 32-Bit-Flex-Architektur
– soft- und hardwarekompatibel zum Industrie-Standard (ISA)
– 80387-Koprozessor mit 25 MHz Taktrate als Option
– 3167-Weitek-Koprozessor mit 25 MHz Taktrate als Option

Arbeitsspeicher
– 1 MByte 32-Bit-RAM (Standard)
– modular erweiterbar bis 16 MByte

Laufwerke:
– 1,2-MByte-Disketten-Laufwerk 5,25" (Standard und optional)
– 1,44-MByte-Disketten-Laufwerk 3,5" (optional)
– 60-MByte-Festplatte oder
– 110-MByte-Festplatte (Zugriffszeit < 30 ms) oder
– 300-MByte-Festplatte (Zugriffszeit < 20 ms)
– externe Festplatteneinheit für zusätzlich ein oder zwei 300-MByte-Festplatten (optional)
– 135-MByte-Bandlaufwerk oder
– 40-MByte-Bandlaufwerk zur Datensicherung (optional)

Farbmonitor (optional):
– VGA-kompatibel, kann bis zu 256 Farben in einer Auflösung von 320×200 Punkten und 16 Farben in einer Auflösung von 640×480 Punkten darstellen

Schwarzweißmonitor (optional):
– Kann Grafik in einer Auflösung von 640×480 Punkten und Text in einer Auflösung von 720×400 Punkten darstellen. Zeigt Buchstaben schwarz auf weißem Grund an.

Steckplätze:
– ein 32-Bit-Steckplatz für den Arbeitsspeicher (belegt)
– fünf 8/16-Bit-Steckplätze, volle Länge
– ein 8-Bit-Steckplatz, volle Länge
– ein 8-Bit-Steckplatz, kurzer Steckplatz

Tastatur:
– COMPAQ-Enhanced-Tastatur mit 102 Tasten
(getrenntes Zahlen- und Cursor-Feld mit zusätzlicher ENTER-Taste, 12 Funktions-Tasten)

Standardschnittstellen:
– eine parallele Schnittstelle für Drucker, 25polig
– eine serielle Schnittstelle für Kommunikation, 9polig

System-Software:
– COMPAQ Expanded Memory Manager (CEMM)
Software-Produkte mit LIM-Spezifikation (Lotus/Intel/Microsoft-Expanded Memory-Spezifikation) können direkt im COMPAQ-Arbeitsspeicher (RAM) verarbeitet werden (max. 8 MByte).
– Disk-Cache-Programm zur Optimierung von Festplatten-Zugriffen
– VDISK (RAM-Disk-Funktion)

Betriebssysteme:
– MS-DOS Version 3.3 (Option)
max. Größe einer logischen Festplatte 512 MByte
– MS OS/2 Version 1.0 (Option)

Leistungssteigerung des P 8000 durch verbesserte WDC-Firmware

Die vom Hersteller ausgelieferten Versionen der WDC-Firmware lassen mittlere Datenübertragungsraten bis maximal 30 KByte/s zu. Dabei besteht eine direkte Abhängigkeit zwischen der Position (Zylindernummer) auf der Winchester-Disk (WD) und der Zugriffszeit der WD. Bei höheren Zylindernummern sinkt die Datenübertragungsrate spürbar (um das 2–3fache) ab!
Die WDC-Firmware wurde so überarbeitet, daß positionsunabhängig (für alle Zylinder der WD) die mittlere Datenübertragungsrate auf zirka 68 KByte/s gesteigert wurde. Dadurch erhöht sich die Leistungsfähigkeit des Gesamtsystems P 8000 um etwa 30–50 Prozent. Die Modifizierung der WDC-Firmware ist für alle mit dem P 8000 ausgelieferten WD möglich.
Der Aufwand zur Modifizierung des WDC beschränkt sich auf das Sichern der auf der WD vorhandenen Nutzerdateien (des /z-Nutzerdirectory), das Umprogrammieren der 2 WDC-Firmware-EPROMs und das Neueinrichten der WD.

PSF 25 957, BfN, Halle, 4002; Tel. 477 333 bzw. 477 335

Görlach/Herrmann

Entwicklungshilfensystem HILFEn für LC-80

Zur Erleichterung bei der Entwicklung von Maschinenprogrammen mit dem Lerncomputer LC-80 wurde das Programmsystem *HILFEn* mit folgenden Funktionen geschaffen:
- Verschieben (Kopieren) von Speicherbereichen
- Automatische Neuadressierung der in Frage kommenden absoluten Adressierung (Absolutsprung, UP-Aufruf, Ladeadresse von Registern) im zu entwickelnden Programm auf Grund der Verschiebung einzelner Programmteile, Tabellen oder des Gesamtprogramms (mit Hilfe einer Verschiebeadreßtabelle, die sich ständig selbst aktualisiert)
- Programmeingabe mit *symbolischen Adressen* (16-Bit-Hexadezimalzahl) wird mit Hilfe der Verschiebeadreßtabelle möglich. Ein Konvertierungsprogramm wandelt alle symbolischen Adressen in absolute Adressen, wodurch ein lauffähiges Maschinenprogramm entsteht.
- Nach Eingabe einer alten Adresse wird die neue Adresse angezeigt (nach Speicherbereichsverschiebung).
- Berechnung von Relativsprüngen mit gleichzeitigem Eintrag in das zu entwickelnde Programm
- Zahlenkonvertierung: Hexadezimal- in Dezimalzahl und umgekehrt
- Hexadezimale Addition/Subtraktion im 16-Bit-Format mit Kettenrechnung und Zahlenkonvertierung
- Vergleich (byteweise) zweier Speicherbereiche
- Suchprogramm einer beliebigen Bytefolge
- Prüfsummenberechnung
- Löschen von Speicherbereichen (Laden mit FFH).

Das Programm befindet sich in einem EPROM (U 2716) und arbeitet zusammen mit dem LC-80-Monitorprogramm. Es ist anhand der Dokumentation leicht zu handhaben, arbeitet *menügesteuert* und ist weitgehend gegen unbeabsichtigte Fehlbedienung gesichert. Außer der Verschiebeadreßtabelle wird der LC-80-RAM von *HILFEn* nicht belegt.

VEB Mikroelektronik „Karl Marx“ Erfurt, Stammbetrieb, Werkteil Nord, Wermutmühlenweg 1, Erfurt, 5066; Tel. 71 34 02/58

Möslein

Hard- und Software für KC 87

Für den Einsatz der Kleincomputer KC 85/1 und KC 87 bei der Lösung wissenschaftlich-technischer Aufgabenstellungen, der Laborautomatisierung und der Textverarbeitung wurde eine Reihe von Zusatzsteckeinheiten sowie speziell darauf abgestimmte Softwarekomponenten entwickelt, über die hier kurz berichtet werden soll:

ADAPTER ZfK 9901
Hierbei handelt es sich um eine Leiterkarte, die den Bus des Kleincomputers bis über die Oberkante des Gehäuses verlängert. Damit wird die Inbetriebnahme und der Test von KC-Steckeinheiten wesentlich erleichtert. Durch Wickelbrücken ist der Adapter auf die Weiterleitung des KC-Busses oder auf die Anpassung an den K 1520-Bus einstellbar.

16 KB DRAM ZfK 9903
Auf einer Leiterplatte der Standardgröße sind 16 KByte DRAM als Speicherbank von 8 × U 256 realisiert.

10 KB SCHALTBARER EPROM ZfK 9905
Häufig werden an einem Kleincomputer entsprechend der verschiedenartigen Aufgabenstellungen mehrere Programmpakete benötigt (z. B. BASIC, ASSEMBLER, BITEX usw.). Es wurde ein schaltbarer EPROM-Modul entwickelt. Durch Ausgabebefehl auf eine festlegbare Adresse kann der Modul zugeschaltet werden; bei Angabe einer alternativen Adresse schaltet er sich ab. Auf diese Weise sind (theoretisch) 11 EPROM-Module (= 110 KByte) am KC gleichzeitig betreibbar.

64 KB DRAM ZfK 9906
Der 64-KByte-RAM-Bereich für das Betriebssystem CP/M wird durch 8 Schaltkreise des Typs U 2164 realisiert (Adreßbereich 4000-E7FF). Durch das Vorhandensein von 16 KByte RAM im Grundgerät wird der Bereich von 4000 bis 7FFF doppelt vergeben. Zwischen diesen beiden 16-KByte-Bänken sowie zwischen normaler Betriebsart und Write-Only kann durch I/O-Befehl umgeschaltet werden.

SERIELLES INTERFACE ZfK 9907
Dieser Modul enthält neben der Adreßkodier- und Verbindungslogik die Peripherieschaltkreise SIO und CTC sowie Interfaceschaltkreise. Durch Wickelbrücken ist IFSS und/oder V. 24 auswählbar.

PARALLEL-INTERFACE ZfK 9902
Auf dieser Steckeinheit sind 2 PIO-Schaltkreise angeordnet. Damit stehen zusätzlich dem User-Port des Kleincomputers 32 Daten- und 8 Handshakeleitungen zur Verfügung. Durch Wickelbrücken können Ausgangssignale durch Treiber verstärkt und Eingangssignale durch Dioden auf den zulässigen Spannungsbereich begrenzt werden.

FLOPPY-DISK-CONTROLLER ZfK 9916
Es wird der Floppy-Disk-Controller U 8272 mit 4 MHz Taktfrequenz eingesetzt. Der Modul unterstützt den Anschluß von Minifloppy-Laufwerken (5,25″) bei doppelter Aufzeichnungsdichte (ein- oder zweiseitig).
Die Steckerbelegung entspricht der des Laufwerktyps K 5600.20 und wird interfaceseitig durch eine Verteilerleiste in Schlitzklemmtechnik realisiert.

Zusatzmonitor ZM
Dieser Monitor steht in einer stark erweiterten Form (gegenüber der ursprünglichen von Robotron) zur Verfügung. Die Erweiterungen betreffen folgende Leistungen:
- Herstellung einer Quasi-CP/M-Kompatibilität
In diesem Betriebsmode können eine Reihe von CP/M-Programmen ohne Änderung auf dem KC abgearbeitet werden (z. B. Turbo-Pascal, ZSID, MBasic etc.).
- Verwaltung der schaltbaren EPROM-Module und des 64-KByte-DRAM-Moduls. Die verdeckte Speicherbank des 64-KByte-DRAM-Moduls wird für die Abbildung direkter (random) Diskettenrufe des CP/M genutzt.
- Neue User-Port-Treiberroutinen für V. 24-DTR-Protokoll (z. B. 1200 und 9600 Baud) und Centronics. Drucker wie Epson LX 86 oder LX 1000 (auch K 6313) sind ohne Druckermodul betreibbar.
- Verbesserte Kassetten-E/A-Routinen
- Umschaltung 20/24-Zeilen-Bildschirmmode.

Turbo-Pascal
Das Programm ist für KC-Bildschirm und Tastatur installiert. Es erfordert den ZM sowie für volle Arbeitsfähigkeit (d. h. Compilieren auch auf Kassette) den 64-KByte-DRAM-Modul. Das Programm ist auch als Kassettenvariante (ca. 30 KByte) verfügbar.

Geräte-Treiberroutinen für den User-Port
SIF 1000-E/A, V. 24-Hardwareprotokoll, Centronics für ZM-unabhängigen Betrieb sowie *Grundsoftware für den SIO-Modul*. Für den SIO-Modul existiert ferner ein Treiberpaket für die Kopplung einer Schreibmaschine (S 6010) im bidirektionalen Betrieb.

CP/M 2.2-kompatibles Betriebssystem
Hardwarevoraussetzung dafür sind der Floppymodul und der 64-KByte-DRAM-Modul. Es sind alle CP/M-Programme arbeitsfähig, die für einen 40 Zeichen breiten Bildschrim installiert sind.
Die genannten Steckeinheiten wie auch die Software stehen Interessenten zur Nachnutzung zur Verfügung. Als Grundleistung werden eine unbestückte Steckeinheit mit Schalt- und Belegungsplan sowie Bedienungsanleitung bereitgestellt.
Die Software ist als Kassetten- oder EPROM-Version (s. o.) lieferbar, wobei das notwendige Material bereitgestellt werden muß. Die lauffähige Software sowie die zugehörige Dokumentation gehören zum Lieferumfang. Darüber hinausgehende Leistungen bedürfen der Vereinbarung.

Zentralinstitut für Kernforschung Rossendorf, Abt. KFM, Postfach 19, Dresden, 8051

Dr. Schwarzenberg/Dr. Fromm

Wir suchen

... ein EPROM-Programmiergerät ab 2716 oder nachnutzbare Dokumentationen.
Schnittstellen für Rechneranschluß IFSS oder V. 24 (DEE) möglichst mit Software für A 7100.

VEB Elektro-Physikalische Werke Neuruppin, Erich-Dieckhoff-Straße, Neuruppin-Treskow, 1951; Tel. 6 14 56

Dr. Bernhardt

... Hard- und Software zur Kopplung des Kleincomputers KC 85/3 mit dem tschechischen Kleinplotter Graphic unit XY 4131 oder dem Robotron-Plotter K 6411. WE.

Hochschule für industrielle Formgestaltung Halle, Burg Giebichenstein, Wissenschaftsbereich Designmethodik, Neuwerk 7, Halle, 4020; Tel. 85 00

Frick

... eine Hard- und Softwarekopplung einer Kassettenmagnetbandeinheit K 5261 mit einem Arbeitsplatzcomputer AC 7150.

VEB Volltuchwerke Crimmitschau, Werk 4 Werdau, August-Bebel-Straße 89, Werdau, 9620; Tel. 24 51

Salomon

... als Betrieb nach einer Lösung, die es ermöglicht, einen Seriendrucker SD 1152 mit IFSS-Anschluß (Bj. 1985) an einem AC A 7150 mit dem Betriebssystem DCP Vers. 3.2. zu betreiben.

VEB Fahrzeuggetriebewerke „Joliot-Curie“ Leipzig, 7030, PSF 144; Tel. 3 95 30

Lange

KC 85/3 im V.24-Verbund für Computerkabinette

Zur Verbesserung der Ausbildung in Computerkabinetten mit KC 85/3 wurde eine Möglichkeit geschaffen, alle Kleincomputer über V.24-Interface zu koppeln. Der Lehrercomputer wurde mit einer Verteilerschaltung versehen, die es gestattet, auf alle Schülercomputer zuzugreifen. Dabei ist es dem Lehrercomputer möglich, unabhängig vom Schülerarbeitsplatz den Arbeits- bzw. Bildwiederholspeicher in dem Lehrercomputer zu lesen bzw. dessen Speicherinhalt in den Schülercomputer zu übertragen. Die Übertragungszeiten sind auf Grund der Bitrate von etwa 54 kBaud kurz. Die Schaltungsunterlagen, die Software und ggf. die Leiterplatte für den Lehrercomputer können nachgenutzt werden.

Betriebsschule „Gustav Meyer" des VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Eisenacher Str. 40, Mühlhausen, 5700; Tel. 53283

Schiwon/Kirves

TDABA

TDABA ist ein schnelles, relational angelegtes Datenbanksystem, das eine Verwaltung von maximal 4 MByte großen Dateien gestattet. Die Menüsteuerung wird über eine eingebaute Window-Technik realisiert und ist so aufgebaut, daß Fehlhandlungen auf Grund von eindeutigen Vorgaben weitgehend ausgeschlossen werden können.
Der Nutzer baut sich eine maßgeschneiderte Bildschirmmaske unter Zuhilfenahme des integrierten Maskeneditors selbst auf und kann die Daten so bequem eingeben. Dabei findet eine Typüberprüfung statt.
Zugelassene Datentypen sind: **STRING, BYTE, INTEGER, REAL**, Festkommazahlen, **CHAR** und **BOOLEAN**.
Ein späteres Ändern der Eingabemaske, also während des laufenden Datenbankbetriebs, ist möglich. Dadurch wird der Nutzer in die Lage versetzt, bereits existierenden und mit **TDABA** verwalteten Daten ein anderes Maskenbild zu geben und mit diesen Daten auch weiterhin zu arbeiten.
Eine Indexbildung zur Unterstützung einer schnellen Suche im Datenbestand kann vom Nutzer wahlfrei vorgenommen werden. Der Index selbst wird im Speicher verwaltet, so daß auch bei größeren Datenbeständen ein schneller Zugriff gewährleistet ist.
Der mit **TDABA** verwaltete Datenbestand kann formatiert auf Bildschirm, Drucker oder in ein File ausgegeben werden. Bestimmte definierte Auswahlkriterien ermöglichen ein Festlegen des äußeren Erscheinungsbildes der Ausgabe und gestatten die gezielte Zusammenstellung von Daten des Bestandes.
Die Übernahme von Daten aus anderen Programmen kann dann erfolgen, wenn sie als ASCII-File vorliegen und für diese Daten vorab eine Eingabemaske bereitgestellt wurde.
Weiterhin ist es möglich, numerische Felder einer statistischen Auswertung zu unterziehen, um Aussagen über Erwartungswert, Varianz, Meßgüte und den Korrelationskoeffizienten zu erhalten.

Technische Voraussetzungen: 16-Bit-PC A 7150 unter DCP 3.2 (Kompatible unter MS-DOS)

Modeinstitut der DDR, Abt. CSZ, Brunnenstraße 19–21, Berlin, 1054

Butte

Hard- und Softwarekomponenten für LOTUNET

Im Rahmen der Nachnutzung bietet der IfAM Erfurt Komponenten des Mittelklasse-LANs LOTUNET des Informatik-Zentrums der TU Dresden zur Vernetzung von 8-Bit-Rechnern (PC 1715/BC) an. LOTUNET ist modular aufgebaut, so daß für jeden Anwender eine seinen Bedürfnissen entsprechende optimale Lösung bereitgestellt werden kann.
Variationsmöglichkeiten ergeben sich u. a. bei der topologischen Struktur (wahlweise Stern-, Linien- oder Ringstruktur), bei den Zugriffsverfahren (CSMA/CD, Polling, Token-Bus, Token-Ring und MSAP möglich) sowie bei der Wahl des Übertragungsmediums Koaxialkabel oder LWL).
Die Datenübertragungsgeschwindigkeit beträgt 250 bzw. 500 kBaud. Maximal können 100 Stationen bei einer Länge des Hauptkabels von 1000 m angeschlossen werden. Bei kleineren Systemen kann das lokale Netz ökonomischer ohne Kabelanschlußeinheit betrieben werden, wenn weniger als 16 Stationen angeschlossen sind und die Länge des Hauptkabels ≦ 500 m ist.
Das Angebot umfaßt neben der Dokumentation industriell gefertigte unbestückte Leiterplatten, umfangreiche Test- und Inbetriebnahmesoftware sowie die Einbindung der Basiskommunikationssoftware in das Betriebssystem des Nutzers.
Weiterhin stehen auf höherer Ebene folgende Kommunikationsdienste zur Verfügung:
- zentrale und verteilte Datenbank
- elektronische Mitteilungssysteme
- Filetransferdienst.

Der Anschluß des EC 1834 an das lokale Netz steht für den Nutzer 1989 zur Verfügung.

VEB Mikroelektronik „Karl Marx" Erfurt, Bereich CI-IFAM, PF 194, Erfurt, 5010; Tel. 6 21 02/18

Dr. Conrad

Abrechnungssystem für DCP

Die **CAMARS**-Gruppe bietet allen Nutzern von DCP oder kompatiblen Betriebssystemen ein komfortables Abrechnungssystem für den Rechenzeitnachweis zur Nachnutzung an. Das System ist für Computer konzipiert, zu denen nur ein fester Personenkreis Zutritt besitzt (zirka 25 Dauernutzer). Voraussetzung ist die Ausstattung des Rechners mit einer Festplatte, die als Laufwerk C: bekannt sein muß.
Leistungsumfang:
* Anmelden der Nutzer, Abarbeitung einer systemspezifischen Startroutine (z. B. Einstellen des Arbeitsdirectory als spezielles Laufwerk)
* Abmelden, Abarbeiten einer systemspezifischen Enderoutine
* Anzeige, Editieren und Löschen von Nutzerspezifikationen
* Anzeige der täglichen Belegung für den laufenden und den vergangenen Monat
* Anzeige der Monatsauslastung für die letzten zwölf Monate
* Druck oder Erstellung eines druckbaren Files für die nutzerbezogene monatliche Auslastung (K 6313-orientiert).

Technische Universität Dresden, Informatikzentrum des Hochschulwesens, WB Angewandte Informatik, Kolln. Grundmann, Mommsenstraße 13, Dresden, 8027

Doetzkies

Nutzung von dBase-Indexdateien unter Turbo-Pascal

Die Verwendung von dBase-Datenbanken in Turbo-Pascal Programmen ist weit verbreitet. Eine effektive Nutzung vor allem großer Datenbanken (>1000 Datensätze) ist in vielen Fällen nur möglich, wenn auch Indexdateien in die Datenbankarbeit einbezogen werden. Dieses Problem wurde mit der Entwicklung der dBase-Toolbox **DBTOOLS** gelöst. Sie bietet dem Nutzer die Möglichkeit des schnellen und direkten Zugriffs auf dBase III– bzw. dBase III+-Datenbanken (Lesen von Datensätzen/Aktualisieren von Datensätzen/Schreiben von Datensätzen) unter Nutzung von dBase-Indexdateien in Turbo-Pascal-Programmen ab Turbo-Pascal-Version 4.00.
Die Toolbox läßt sich als Unit in beliebige Anwenderprogramme einbinden und ermöglicht durch die Indexdateinutzung, große Datenmengen mit kleinen Zugriffszeiten zu bearbeiten. Bei der Festlegung des Funktionsumfangs der Toolbox wurde vor allem auf die universelle Datenbanknutzung geachtet. So ist es möglich, gleichzeitig drei Datenbanken zu bearbeiten, wobei jede Datenbank wahlweise
- in Verbindung mit einer Indexdatei
- durch sequentielle Bearbeitung ohne Indexdatei
- durch Direktpositionierung auf beliebige Datensätze genutzt werden kann.

Die dBase-Toolbox DBTOOLS kann in Verbindung mit einer umfangreichen Dokumentation, in der u. a. anhand mehrerer Anwendungsbeispiele die Arbeit mit der Toolbox erläutert wird, nachgenutzt werden.

Friedrich-Schiller-Universität Jena, Sektion Technologie für den WGB, Wissenschaftsbereich Prüftechnik, Ernst-Thälmann-Ring 32, Jena, 6900

Orth

Umwandeln von SCP-Dateien in DCP 1700-Dateien

Wir bieten ein Programmsystem zum Umwandeln von SCP- in DCP 1700-Dateien (und umgekehrt) für den Arbeitsplatzcomputer A 7150 an. Es besteht aus je einem in SCP und in DCP auf dem gleichen Rechner zu startenden Programm. Eine Rechnerkopplung entfällt hierbei.
Da auf diesem Rechner mit Hilfe des SCP 1700-Kommandos DISKSET verschiedene Diskettenformate gelesen werden können, lassen sich auch SCP-Dateien, die auf anderen Rechnertypen erstellt wurden, in DCP-Dateien umwandeln.

Institut für Schiffbautechnik, Am Strom 109, Rostock-Warnemünde, 2530; Tel. 5 62 76

Herrmann

Ausdruck von Zeichnungen auf Nadeldruckern

Mit wachsendem Einsatz von CAD-Software steigt der Bedarf an grafischer Ausgabe von Zeichnungen. Oft reichen die vorhandenen Plotter nicht aus, oder sie erfordern einen zu hohen Bedienungs- und Beaufsichtigungsaufwand. Diesem Mangel kann das Printplot-Programm **PRIPLO** entgegenwirken. Es wurde im Rahmen eines Neuerervorschlages im ZKI realisiert und ermöglicht das Plotten von Zeichnungen auf 16-Bit-Rechnern mit epsonkompatiblen Druckern (z. B. K 631x). Folgende grafische Grundsymbole können verwendet werden:
- Linien
- Maßpfeile
- Rechtecke
- Kreise, Kreisbögen
- Marker
- Texte
- Schraffuren

(mit Einschränkungen).
Die Plotteranweisungen befinden sich in einem Textfile und können in einem beliebigen xy-Koordinatensystem programmiert sein. Das Bild wird automatisch auf ein passendes Format transformiert und mittig platziert.
Die Bedienung ist sehr einfach und wird durch eine Help-Funktion sowie ein ausführliches Demonstrationsprogramm unterstützt.

Akademie der Wissenschaften, Zentralinstitut für Kybernetik und Informationsprozesse, Kurstraße 33, Berlin, 1086; Tel. 2 07 22 32

Dr. Zorn

Bildschirmorientierter Editor BiTEX für P8000

Der bei vielen Nutzern der 8-Bit-Systeme MPS 4944 und K 1520 angewendete bildschirmorientierte Editor BiTEX wurde für das P8000-Terminal und die Betriebssysteme UDOS sowie OS/M adaptiert. Nach Erfahrung beim Verfasser kann die Arbeitseffektivität bei Programmerstellung und -modifikation gegenüber den Zeileneditoren ‚ED' etwa verdoppelt werden. Die Erlernung des Editors ist denkbar einfach: Alle Textveränderungen werden sofort angezeigt, einige Kommandos erleichtern blockorientierte und File-Arbeit.

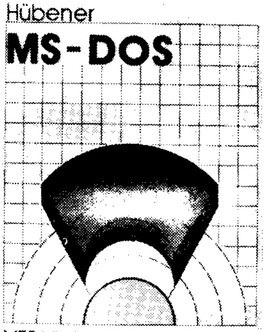
ZfK Rossendorf, Abt. Woe, PSF 19, Dresden, 8051; Tel. 5 91 20 84

Thomae

MS-DOS

von J. Hübener, VEB Verlag Technik, Berlin: 1988, 240 S., 2 Bilder, 67 Tafeln, Broschur, DDR 24,– M, Bestell-Nr. 554 097 5





Wer mit einem IBM-PC (oder einem kompatiblen Rechner) unter einem MS-DOS-artigen Betriebssystem arbeitet und trotz vieler nutzungsfertiger Softwarepakete spezielle eigene Programme benötigt, der steht vor einer schier unüberschaubaren Menge von Handbüchern und Spezialliteratur. Der Autor des vorliegenden MS-DOS-Bandes der Reihe Technische Informatik hatte sich vorgenommen, „in komprimierter Form eine Einarbeitung in MS-DOS zu unterstützen und durch Beschreibung der internen Zusammenhänge das Verständnis für die Arbeitsweise des Systems zu erleichtern". Das ist ihm sehr gut gelungen. Das Schwergewicht liegt auf der Darstellung der BIOS- und DOS-Funktionen: Wie ruft man sie beispielsweise aus PASCAL-, C-, FORTRAN- und Assemblerprogramm auf? Es ist also kein Buch für Programmieranfänger. Ein bereits mit irgendeiner Form von Rechentechnik vertrauter Leser, der sich die MS-DOS-Welt auf 16-Bit-Rechnern erschließen will, sollte nach der Lektüre der Kapitel 1 und 2 („Einführung", „PC-Hardware", 15 Seiten) zunächst große Teile der zweiten Hälfte des Buches studieren: Kapitel 6 „DOS-Kommandos" (40 S.), 7 „Stapelverarbeitung" (7 S.), 8 „Werkzeuge zur Softwareentwicklung" (20 S.) und 10 „Anwenderprogramme" (15 S). Wer danach auch durch die kurzen Abschnitte 9 „Alternative Benutzerschnittstellen" (6 S.) und 12 „Alternativen zu MS-DOS" (2 S.) nicht von seinen Absichten zur system- und hardwarenahen Programmierung abzubringen ist, der sollte insbesondere den Abschnitt 3.1 „Programmiermodell" (6 S.) genau lesen. Anschließend findet man in den Kapiteln 4 „Struktur von MS-DOS" (50 S.) und 5 „Dateisystem" (30 S.) sowie im Anhang (12 S.) die benötigten Angaben zu den BIOS- und DOS-Funktionen. Die „Praktischen Tips und Programme" (Kap. 11, 10 S.) liest man am besten parallel dazu.

Prof. Dr. Hans Schiemangk

Schaltungssammlung für den Amateur

Vierte Lieferung – 1. Auflage
von K. Schlenzig u. a., Berlin: Militärverlag der DDR, 1986, 200 S., 365 Bilder, DDR 16,– M

Alle Leser der nunmehr vierten Lieferung der Schaltungssammlung – ob Amateur, Werkstattpraktiker oder Funker – finden in dieser Sammlung wiederum wertvolle Hilfe.
Durch den kürzeren Zeitraum zwischen den jeweiligen Lieferungen sind auch günstigere Bedingungen für das vorgesehene Anknüpfen an Vorangegangenes geschaffen. Das ist auch notwendig, um dem Entwicklungstempo insbesondere der Mikroelektronik entsprechen zu können.
Die vorliegende Sammlung bietet wieder eine Fülle von Schaltungen. Dabei sind diese bei Bedarf variierbar und können auch teilweise miteinander verknüpft werden. Bei vielen Schaltungen ist ein Nachbau durch vorgegebene Platinen-Layouts erleichtert.
Die schon seit der 2. Lieferung nicht mehr mögliche Themenbegrenzung auf jeweils ein Blatt brachte wieder die Verteilung großer Themen auf mehrere Arbeitsblätter.
Aus Anschlußgründen werden im Kapitel „Mikroprozessortechnik" auf 5 Blättern noch die Arbeiten zum ZE 1 auf U 808-Basis beendet. Das entspricht der dynamischen und rasanten Entwicklung der Mikroelektronik, die schneller war als die Erscheinungsfolge der Schaltungssammlungen. Mit 15 Blättern widmet sich die Sammlung dann aber dem weit verbreiteten, auch schon seit langem in der Hand des Amateurs befindlichen Prozessor U 880D. Mit insgesamt 20 Blättern hat damit die Mikroprozessortechnik den ihr zustehenden gewichtigen Platz in dieser Sammlung erhalten. Dabei werden hier sowohl die theoretische als auch die applikative Seite der neuen Technik gleichzeitig behandelt.
Auch alle anderen Kapitel vermitteln bei reinen Hardwarelösungen interessante Anregungen auf unterschiedlichste Art und Weise. Mit der vierten Lieferung der Schaltungssammlung ist wiederum eine Arbeitsgrundlage für manchen praktisch Tätigen gegeben.

Dr. I. Schreiber

Zeitschriftenschau

Grafik für die SM-4

Von der polnischen Firma MERCOMB wurde ein spezielles Grafiksystem für die ansonsten nicht grafikfähigen Computer der Serie SM-4 und MERA 60/600 geschaffen. Unter dem Namen KOLOGRAF K-640/QK-640 wird die entsprechende Hardware (Steuereinheit, hochauflösender Grafikmonitor) und Software angeboten. Damit werden die genannten Computer u. a. CAD-fähig.
Anschrift der Firma für Nachfragen: CBW „MERCOMB" Sp.z. o. o. ul. Poezji 19, 04–994 WARSZAWA

aus KOMPUTER (1988) S. 45

Transputer KMAX für den ATARI ST

Die britische Firma KUMA bietet seit einiger Zeit einen Transputerzusatz für den ATARI ST mit dem 32-Bit-Mikroprozessor INMOS T 414 und 256 KByte RAM an. Die Taktfrequenz beträgt 15 MHz und soll auf 20 MHz erhöhbar sein. Zusammen mit der Hardware werden dem Transputernutzer noch das Handbuch und eine Diskette mit den nötigen Programmen zum Installieren des Transputers angeboten. Neben dem Installationsprogramm befinden sich im Programmpaket *INMOS Transputer Developement System* noch ein OCCAM II-Compiler sowie eine Reihe weiterer Hilfsprogramme.
Dieser Zusatz ermöglicht den effektiven Einsatz des Computers u. a. bei Echtzeitproblemen, im CAD/CAM, bei umfangreichen wissenschaftlich-technischen Berechnungen und in der Computergrafik.

aus KOMPUTER (1988) 1, S. 38

Mikroprozessor K 1801 WM 1 aus der UdSSR

Der K 1801 WM 1 entspricht dem Prozessor des Mikrorechners LSI-11 der Firma Digital Equipment Corporation in einer Einchipausführung. Die Versorgungsspannung beträgt 5 Volt, die Taktfrequenz 5 MHz, Fertigungstechnologie NMOS, 16-Bit-Worte, die Anzahl der Befehle beträgt 64. Die Operationszeiten liegen zwischen 2 und 10 Mikrosekunden.
Der Nachfolgetyp K 1802 WM 2 ist doppelt so schnell und besitzt 8 weitere Befehle, die die Arithmetik, einschließlich Fließkommaarithmetik, erweitern.

aus KOMPUTER (1988) 2, S. 10

Computer in der UdSSR

Der seit 1985 in der UdSSR im Angebot befindliche Heimcomputer Elektronika BK 0010 hat einen Nachfolger erhalten, den BK 0010.10. Ausgerüstet ist er mit einem 16-Bit-Mikroprozessor K 1801 WM 1. Standardmäßig ist er mit 64 KByte Speicher ausgerüstet, RAM und ROM teilen sich den Speicherplatz zu gleichen Teilen. Die Tastatur besteht aus 74 Tasten, darunter eine vergrößerte RETURN-Taste und ein Extrablock für die Cursorsteuerung. Die Belegung der Tasten entspricht der sowjetischen Norm. Verfügbar sind der ASCII- und russische Zeichensatz.
Das Gerät verfügt über folgende Anschlüsse:
Stromversorgung 5 V, Peripheriegeräte, RGB, Video, Magnetbandgerät.
Der Bildspeicher benutzt 16 KByte des RAM, so daß 16 KByte für die Programme zur Verfügung stehen. Der RAM-Bereich ist umschaltbar, es stehen dann nur 4 KByte für den Bildwiederholspeicher, aber 28 KByte für Programme bereit. Ein Gerätetestprogramm steht neben einem Monitorprogramm auf einer Zusatzkarte zur Verfügung. Ein komfortables BASIC (ähnlich dem MSX-Standard) ist im ROM auf EPROM vorhanden und meldet sich nach dem Einschalten des Computers.
Als externer Massenspeicher ist bisher ausschließlich die Magnetbandkassette vorgesehen, die Datenübertragung erfolgt mit etwa 1200 Baud. Seit einigen Monaten werden in „Elektronika"-Geschäften Kassetten mit Programmen zum Kauf angeboten. Für den Anschluß eines normalen TV-Empfängers ist ein TV-Modulator extra zu erwerben.

aus KOMPUTER (1988) 2, S. 11–12

Kurz vorgestellt: Bondwell 8

Der Computer besitzt einen CMOS-80C88-Prozessor, der mit einem Takt von 4,77 MHz arbeitet. 512 KByte RAM und 8 KByte ROM stehen in der Grundversion zur Verfügung. Als externer Massenspeicher werden 3,5"-Disketten mit einer Kapazität von 720 KByte benutzt. Der LCD-Bildschirm hat eine Auflösung von 640 × 200 Punkten. Über 76 Tasten ist der Computer bedienbar. Für die Verbindung mit der Außenwelt stehen ein Centronics- und ein RS-232-Interface zur Verfügung. Ein zusätzliches Diskettenlaufwerk ist anschließbar. Weitere Anschlüsse gestatten die Benutzung eines s/w-Monitors und eines Farbmonitors (RGB). Als Betriebssystem wird die Version des MS-DOS 2.11 angeboten, BASIC steht als GW-BASIC V 2.0 zur Verfügung. Die Stromversorgung des Portables wird durch Akkumulatoren 12 Volt/3 Ah bzw. durch ein externes Netzteil mit 13,2 Volt (1,4 A) realisiert. Gewicht ca. 4,5 kg.

aus KOMPUTER (1988) 3, S. 40 ff

MS-DOS in historischer Sicht

In der KOMPUTER-März- und Aprilausgabe wird über die Entwicklung des Programmsystems MS-DOS in einem Vergleich der Versionen 1.0 bis 3.3 berichtet. Dabei werden die wichtigsten Veränderungen (an denen die rasante Entwicklung der Computerwelt ablesbar ist) zu jeder Version dargestellt. Eine Tabelle der Länge der zum System gehörenden Files zeigt die quantitativen Veränderungen. Des weiteren wird die der Klasse PC/XT zugehörige Version 2.0 und 2.1 etwas genauer betrachtet.

aus KOMPUTER (1988) 3, S. 25–26 und 4, S. 36–37

Virusinfektion

Wie „KOMPUTER" im Aprilheft meldet, wurde die Redaktionsarbeit durch einen eingedrungenen Computervirus gestört. Nach den mitgeteilten Angaben handelt es sich um ein etwa 600 Byte langes Exemplar, das sich vor allem an COM-Files anlagert und diese zerstört. Als einziges Gegenmittel wurde der regelmäßige Test der vorhandenen Programme empfohlen.

aus KOMPUTER (1988) 4, S. 4

32-Bit-Prozessor mit flexibler Architektur

Die japanische Firma VM Technology Corp. hat einen 32-Bit-Mikroprozessor entwickelt, der eine ganze Prozessorfamilie von Intel und NEC ersetzen soll. Um Copyrightprobleme zu vermeiden, wird auf den Einbau eines Mikrocodes verzichtet und eine virtuelle Prozessorarchitektur verwendet. Die flexible Architektur erlaubt den späteren Einbau von Instruktionsets nach vorhandenen Industriestandards mittels programmierbarer Logikarrays. Der Hersteller nennt dies anwenderspezifische integrierte Prozessoren.
In Entwicklung ist ebenfalls eine 16-Bit-Version. Die neuen Prozessoren sollen billiger als die Originale von Intel und NEC* sein und mit ihrer Verwendbarkeit für MS-DOS-Computer möglicherweise den Bau von kompatiblen XT- und AT-PCs erlauben. *MP*

KI-Workstation mit 40-MHz-Mikroprozessor

Als „schnellste Workstation", die jemals entwickelt wurde" hat Texas Instruments die Workstation *Explorer II Plus* vorgestellt.
Sie ist als Workstation für symbolische Datenverarbeitung speziell für die Entwicklung von umfangreichen und komplexen wissensbasierten Systemen konzipiert. Einsatzschwerpunkte sind Aufgaben der künstlichen Intelligenz (KI) in großen Unternehmen, Forschung und Entwicklung und anderen Bereichen, in denen die Leistungsfähigkeit und Technologie einer optimierten KI-Workstation erforderlich sind. Die Workstation basiert auf der neuesten Version des Explorer Lisp Mikroprozessors von TI, welcher der erste speziell für KI-Anwendungen konzipierte 32-Bit-VLSI-Mikroprozessor ist. Er ist in 1,0-Micron-CMOS hergestellt; auf einer 1 Quadratzentimeter großen Fläche sind mehr als 553.000 Transistoren untergebracht. Diese Technologie ermöglicht eine Eingangs-Taktfrequenz von 80 MHz und eine Mikrobefehls-Taktrate von 40 MHz.
Wie die bisherigen Explorer-Modelle, so basiert auch der Explorer II Plus auf einer prozessorunabhängigen 32-Bit-NuBus-Architektur. Explorer II Plus-Systeme sind mit Hauptspeicherkapazitäten bis zu 128 MByte lieferbar. Zu den Massenspeichereinheiten gehören 5 1/4-Zoll-Bandkassettenlaufwerke und 1/2-Zoll-Magnetbandlaufwerke. Die Systemkonsole ist mit hochauflösendem (1024 × 808 Pixel), monochromem 17-Zoll-Bildschirm oder 8-Bit-Farbbildschirm mit 17-Zoll-Diagonale lieferbar. Für jede beliebige Explorer-Konfiguration steht ein Prozessor auf 68020-Basis mit dem TI-System V-Betriebssystem zur Verfügung.
Mit diesem Prozessor läßt sich das System zu einer LX-Version aufrüsten, die eine Integration von konventionellen Softwareprogrammen mit KI-Anwendungen ermöglicht.
Das Foto zeigt den Explorer II Plus mit Monochrombildschirm (links) und Farbmonitor (Mitte) sowie den microExplorer, der die Leistung des Lisp-µP mit den Leistungen der Macintosh II-PCs verbindet und zu dem die neue Workstation softwarekompatibel ist. *MP*

Laserdrucker mit 600 dpi

Zur Orgatechnik im Herbst 1988 stellte die Firma Compugraphic ihren neuen grafikfähigen Laserdrucker CG 610 vor, der eine Auflösung von 600 Punkten pro Zoll (dots per inch-dpi) erreicht (als Standard bei Laserdruckern gelten gegenwärtig 300 dpi). Aufgrund der damit möglichen 360 000 Punkte pro Quadratzoll ist der CG 610 nicht nur zum Anfertigen von Prüfausdrucken (Proofprints) in der Polygraphie geeignet, sondern bereits für Endprodukte.
Der Laserdrucker hat eine maximale Ausgabeleistung von 10 Seiten pro Minute und verarbeitet A4-Normalpapier. Durch das identische Raster Image Control System (ICS) ist er mit den gleichen Parametern ansteuerbar wie der neue Laserbelichter von Compugraphic, der CG 9400. *MP*

Super-VGA

Von der Firma NEC wurde auf der amerikanischen Computermesse Comdex im Herbst 88 eine Interessengemeinschaft gegründet, die eine einheitliche Behandlung der Bildschirmauflösung von 800 × 600 Bildpunkten propagiert. Erste Mitglieder sind, neben NEC, ATI, Genoa, Orchid, STB, Tecmar, Video-7 und Paradise. Auf dieser „Super-VGA"-Basis könnte ein Standard entstehen, auch wenn IBM nicht beteiligt ist. *MP*

Schneller Matrixdrucker von Seikosha



Mit einer Geschwindigkeit von 800 Zeichen je Sekunde ist der SBP-10 von Seikosha der wohl noch immer schnellste Nadeldrucker – zumindest bei den mit nur einem Druckkopf ausgerüsteten Modellen. Die mit zwei oder drei 9-Nadel-Druckköpfen ausgestatteten – und evtl. schnelleren – Printer können ihre Druckgeschwindigkeit im Gegensatz zum SBP-10 jedoch nur bei voller Druckbreite ausspielen. Der 18-Nadeldruckkopf ist laut Hersteller der kleinste und leichteste der Welt. Die Masse des Druckkopfes spielt bei dieser Druckgeschwindigkeit eine große Rolle, werden doch pro Sekunde die Nadeln insgesamt etwa 60 000mal abgefeuert. Der Druck erfolgt bidirektional mit Druckwegoptimierung.
Als weitere Besonderheiten des Druckers werden seine – trotz der Schnelligkeit – geringe Geräuschentwicklung von < 260 d (BA) nach ISO-DIS 7779 und die leichte Bedienbarkeit hervorgehoben. So können sämtliche Funktionen des Druckers über das Bedienfeld angewählt und auf einem zweireihigen LC-Display verfolgt werden. Der Drucker bietet mehrere Möglichkeiten der Papierzuführung, beispielsweise auch von unten. Auch lassen sich sowohl Endlosformulare als auch Einzelblätter verarbeiten. *MP*

PS/2 in China?

Wie die PC-Woche in ihrer Ausgabe 39/88 meldet, plant IBM mit der Volksrepublik China zwei Joint-Venture-Abkommen. Gemeinsam soll ein Kundendienstzentrum aufgebaut und Software entwickelt werden.
Auch sei der Bau eines Werkes zur Herstellung von Computern des IBM-Personal Systems/2 vorgesehen, die mittelfristig im Inland vertrieben werden sollen. *MP*

32-Bit-GaAs-Mikroprozessor

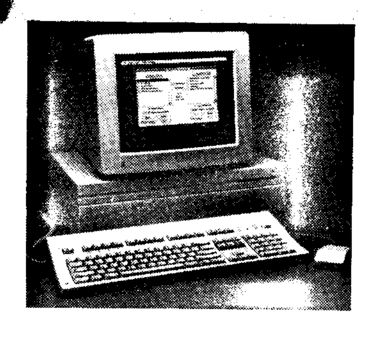
Den nach Aussagen des Herstellers ersten 32-Bit-Galliumarsenid-Mikroprozessor auf RISC-Basis (Reduced Instruction Set Computer) hat Texas Instruments vorgestellt.
Mit 12 895 Gattern dürfte dieser Mikroprozessor der größte funktionale Logikbaustein sein, der jemals in GaAs-Technologie entwickelt und produziert wurde.
Bis Ende 1988 war eine Geschwindigkeit von 100 MHz zu erreichen; als Design-Ziel bis Ende 1989 erwartet TI 200 MHz und 200 Millionen Befehle pro Sekunde. Neue und bisher einmalige Innovationen im Entwurf erhöhen die Geschwindigkeit dieses Mikroprozessors.
Eine 6stufige Pipeline-Architektur ermöglicht den Zugriff auf die Daten noch in der Pipeline und reduziert die Zugriffszeit auf den Speicher. Um die Datenverzögerung in den kritischen Pfaden zu reduzieren, wurden selektiv High-Speed-Gatter verwendet. Die kritische Pfadlänge der CPU umfaßt 30 Gatterverzögerungen. Die Selektion der Gattergeschwindigkeit wird während der Herstellung mit einer programmierbaren Kontaktmaske erreicht.
Laufzeitverzögerungen von 160 ps sind für den kritischen Pfad erforderlich, um eine Zykluszeit von 5 ns zu erreichen, die wiederum Voraussetzung ist für den Betrieb bei 200 MHz. *MP*

16-Bit-µP bald mit 33 MHz?

AMD hat einen 80 286-Prozessor mit 25 MHz Taktfrequenz angekündigt, dem noch in diesem Jahr eine 33-MHz-Variante folgen soll. Damit will man sich offensichtlich gegen den 80 286-Nachfolger 80386SX behaupten. *MP*

Neue Macintosh-Version IIx

Apple Computer hat in San Fracisco den ersten Macintosh mit einem 68 030-Mikroprozessor von Motorola vorgestellt: den Macintosh IIx.
Die Kombination aus 68 030-Mikroprozessor und 68 882-Koprozessor, beide mit 16 MHz getaktet, führt zu einer Leistungssteigerung von 10 bis 15 Prozent gegenüber dem Macintosh II. Die gesamte Macintosh II-Software läuft ohne Modifizierung auch auf dem Macintosh IIx. Der Macintosh IIx wird in zwei Ausführungen ausgeliefert: entweder mit einer 80-MByte-Festplatte und 1,44-MByte-Diskettenlaufwerk oder nur mit einem Diskettenlaufwerk. Beide Versionen verfügen über 4 MByte RAM und ermöglichen den Einsatz anspruchsvoller Anwendungen, da sie die Fähigkeiten des MultiFinders, einer Erweiterung des Macintosh-Betriebssystems, voll ausschöpfen können.
Ebenfalls neu ist das 3,5-Zoll-Diskettenlaufwerk (FDHD – Floppy Drive High Density) mit einer Speicherkapazität von 1,44 MByte für den



Macintosh IIx, das auch MS-DOS, OS/2 und Apple II-Dateiformate lesen und schreiben kann.

MP

Toshiba entwickelt 32-Bit-Mikroprozessor für TRON-Projekt

Die Firma Toshiba entwickelte auf der Basis der Tron-Chip-Architektur einen neuen 32-Bit-Mikroprozessor mit der Bezeichnung TX1, der besonders für die Verwirklichung des TRON-Projektes von Bedeutung sein soll. Dabei soll er in Robotern mit Künstlicher Intelligenz und in schnellen Echtzeitsteuersystemen für Betriebe und Kraftwerke Verwendung finden. Toshiba plant, den TX1-Chip außerdem als Kernprozessor von ASICs einzusetzen.
Die technischen TX1-Muster und die peripheren Chips, einschließlich eines Taktgebers und eines Steuerteiles für den direkten Speicherzugriff, sollen noch in diesem Jahr geliefert werden.
Die durchschnittliche Arbeitsgeschwindigkeit des TX1 wird mit 5 MIPS und die maximale Arbeitsgeschwindigkeit bei einem Taktzyklus von 25 MHz mit 12,5 MIPS angegeben.
Der TX1 umfaßt 450 000 Transistoren auf einer Fläche von 10,89 × 10,27 mm².

ADN Fa

Masscomp: Echtzeit-System auf Unix-Basis

Die Firma Masscomp stellte im Frühjahr dieses Jahres eine neue Reihe von Echtzeit-Rechnern vor, die alle mit dem auf der Basis des Betriebssystems Unix selbst entwickelten Echtzeit-Betriebssystem RTU arbeiten. Bei dieser neuen Reihe handelt es sich um vier System-Linien, die alle binärkompatibel sein sollen, d. h., die alle den gleichen Maschinencode verstehen, und die als Mehrprozessor-Maschinen aufgebaut sind.
Das leistungsstärkste System der neuen Familie mit der Bezeichnung *MC 6700* kann bis auf maximal fünf Prozessoren des Motorola-Typs 68030 ausgebaut werden. Zu jedem dieser Universal-Mikroprozessoren gehören 64 KByte Pufferspeicher und je ein eigener Gleitkomma-Beschleuniger. Werden außerdem noch vier sogenannte Vektor-Beschleuniger eingebaut, könnte dieses System evtl. 56 Mio Gleitkomma-Befehle pro Sekunde (56 MFLOPS) bearbeiten.
Diese Mehrprozessor-Rechner sollen den speziellen Anforderungen ihres jeweiligen Einsatzes außerdem durch den Einbau von maximal vier Ein-Ausgabe-Subsystemen auf Multibus-Basis, von zwei gleichen Systemen auf Basis des VME-Bus sowie von zwei sogenannten Datenerfassungs-Subsystemen angepaßt werden. Die Reaktionszeit wird von Masscomp mit 5 ms angegeben.
Außerdem wurde das Betriebssystem Unix stark modifiziert und erweitert, um die Echtzeitfähigkeit zu erreichen.

Quelle: VDI-Nachrichten vom 10. 6. 1988 Fa

Löschbare Optik-Speicher

Die kalifornische Firma Maxtor Corp. entwickelte einen mit löschbaren Optikplatten arbeitenden Massenspeicher, mit dem man versuchen will, das Problem der bisherigen Laufwerke, die relativ hohe Zugriffszeit und die mäßige Transfergeschwindigkeit, zu lösen.
Das Laufwerk trägt die Bezeichnung Tahiti 1 und arbeitet mit einer Zugriffszeit von 43,5 ms und einer Transferrate von 13,7 MBit/s. Die Speicherkapazität jeder 5,25-Zoll-Optikplatte beträgt maximal 1 GByte. Erste Muster sind noch nicht lieferbar. Das Laufwerk soll für OEM-Kunden für einen Preis von 2500 Dollar erhältlich sein und die Platten für 150 Dollar.
Gleichzeitig entwickelte die Firma Maxtor ein kleineres System Fiji 1, das mit 3,5-Zoll-Platten von 160 MByte Kapazität arbeitet. Die Zugriffszeit beträgt 100 ms und die Transferrate 2 MBit/s.

Quelle: Elektronik. – München 37 (1988) 16. – S. 7 Fa

Farbe auf Weiß

An den verschiedenen Farbdarstellungen bei Flüssigkristall-Displays arbeitet eine Reihe namhafter Firmen. Der japanischen Firma Seiko Epson soll es erstmals gelungen sein, ein Flüssigkristall-Display zu entwickeln, das Farbdarstellungen auf weißem Hintergrund ermöglicht. Als Grundlage dient eine Kombination aus zwei Flüssigkristall-Schichten. Dazu wurde eine Farbfilterschicht, die aus einer Punktmatrix mit den drei Grundfarben besteht, hinzugefügt. Das Display arbeitet passiv, das heißt, ohne eine Matrix aus Steuertransistoren. Damit ergibt sich einerseits ein deutlich geringerer Produktionsaufwand, aber andererseits verlängert sich die Ansprechzeit. Sie liegt beim gegenwärtigen Prototyp (12 cm Diagonale, 240 × 480 Punkte) bei etwa 40 ms.

Quelle: Elektronik. – München 37 (1988) 18. – S. 7 Fa

Neues japanisches Computerprojekt

Ein weiteres Projekt zur Entwicklung von Computern der nächsten Generation soll mit einem Aufwand von mehreren Milliarden Dollar in Japan gestartet werden. Zunächst soll im Zeitraum eines Jahres eine Studie erarbeitet werden, in der die Realisierungsmöglichkeiten für Neuro-Computer, Bio-Computer und Optik-Computer untersucht werden sollen.
Ab März 1989 soll auch ein Sechs- bis Achtjahresprojekt beginnen, das Voraussetzungen für die Fuzzy-Rechentechnik schafft. Die Fuzzy-Rechentechnik basiert auf einer multivariablen und nicht auf der binären Logik.

Quellen: Süddeutsche Zeitung vom 17. 8. 1988; Datamation. – Barrington 34 (1988) 16. – S. 14 Wi

Weitere Ergebnisse der Mikromechanik

Einen Mikromotor von 60 μm Durchmesser und 2 μm Höhe sollen Wissenschaftler der Universität California entwickelt haben. Der Motor wurde mit Hilfe von Ätzverfahren hergestellt, wie sie in der Mikroelektronik gebräuchlich sind. Dabei wurden Polysilikon-Bauteile auf Siliziumdioxid plaziert. Durch das Herauslösen des Stützwerkes erhält man frei bewegliche Scheiben, die Rotoren des Motors. Treibende Kraft des Motors ist die elektrostatische Anziehung. Sie tritt auf, wenn zwischen schlecht leitenden Stoffen eine höhere Spannungsdifferenz auftritt. Die Spannung wandert entlang von Lamellen, die sich rings um den Rotor befinden. Die einzelnen Zähne des Rotors folgen der Spannung, wodurch der Motor in Bewegung gerät.
Für den Motor ist noch die Komplettierung durch Ventile, Federn, Filter und Stecker erforderlich.

Quelle: Geo (1988) 10 Wi

IBM informierte über Entwicklungsrichtungen

Die Firma IBM hat im Zusammenhang mit der Ankündigung von 10 Rechnern der Serie IBM 3090 erstmals auch Informationen über längerfristige Vorhaben bekanntgegeben. Bisher bestand die Strategie der Firma darin, Kunden und Konkurrenten möglichst lange über die nächsten Veränderungen im unklaren zu lassen.
Die wichtigsten Ankündigungen sind folgende:
– Kanäle mit 20 MBit/s für 1989, eine weitere 5fache Steigerung der Kanalgeschwindigkeit bis 1991;
– alle 3 VM-Betriebssysteme werden zu einem einzigen Betriebssystem vereinheitlicht;
– OSI wird bis Ende 1990 in SAA eingeschlossen;
– die Entwicklung von AIX Version 3 als Basis für das Konkurrenzprodukt der Open Software Foundation zu Unix soll 1989 abgeschlossen werden.

Quelle: Computer today. – New Delhi 4 (1988) 43. – S. 14 Wi

Diskettenspeicher großer Kapazität

Mittels eines neuartigen Schreib-/Lese-Verfahrens soll es einer US-Firma gelungen sein, eine 25mal größere Kapazität gegenüber herkömmlichen Disketten zu erreichen. Das Schreib-/Lese-Verfahren besteht aus der Kombination eines Laserstrahles für die Ansteuerung des Sektors und eines konventionellen Magnetkopfes für die Realisierung der eigentlichen Schreib-/Lese-Vorgänge.

Quelle: Die Welt vom 28. 9. 1988

Wi

Pläne für Hochleistungsrechner in Indien

Analog zu den Programmen auf den Gebieten Raumfahrt und Atomenergie haben sich die indischen Wissenschaftler auch für die Rechentechnik ehrgeizige Ziele gestellt.
Seit 1986 gibt es Pläne, parallelverarbeitende Superrechner selbst zu entwickeln. Die Entwicklung von Spitzentechnologie soll insgesamt das Niveau der Eigenproduktion der Rechentechnik erhöhen.
Im Juli 1987 soll das National Aeronautics Laboratory den Prototyp eines Multiprozessorsystems mit 4 Prozessoren (Intel 80386, Intel 80387) vorgestellt haben. Das System erreicht 1,3 MFLOPS. Für 1990 wurde eine Version mit 10 MFLOPS angekündigt. Im Jahre 1988 wurden unter Mitwirkung des Ministeriums für Wissenschaft und Technik weitere Projekte für die Entwicklung hochleistungsfähiger Rechner erarbeitet. Gegenwärtig soll eine 200 Personen umfassende Entwicklungsgruppe aufgebaut werden. Bis Ende 1990 soll ein Parallelverarbeitungssystem mit einer Geschwindigkeit von 1000 MFLOPS entwickelt werden.

Quelle: Computer today. – New Dehli 4 (1988) 43. – S. 30–41 Wi

Silizium für optoelektronische Bauelemente

Siliziumkristalle mit einem Durchmesser von 23 bis 30 Ångström auf einer dünnen Trägerfläche sollen nach Erkenntnissen japanischer Wissenschaftler bei Raumtemperatur rotes Licht aussenden, wenn sie Argon-Laserstrahlen ausgesetzt werden. Der Einsatz von Silizium in lichtaussendenden Dioden für die Optoelektronik anstelle des aufwendig zu produzierenden Gallium-Arsenid wäre damit möglich. Die Dioden kommen in optischen Kommunikationssystemen zum Einsatz.

Quelle: eee. Elektronik-Technologie ... – Leinfelden-Echterdingen (1988) 18. – S. 72 Wi

Automatische Textübersetzung Englisch–Chinesisch

Unter der Bezeichung *Transtar* stellte die chinesische Software Corporation ihr automatisches Übersetzungssystem für Textübertragungen aus dem Englischen ins Chinesische vor. Wie die chinesische Nachrichtenagentur meldet, umfaßt das System mehr als 100 000 englische Wörter, darunter die 40 000 am häufigsten vorkommenden Begriffe. Es erkennt außerdem Begriffe aus dem Fachwortschatz der Computer- und Kommunikationstechnik sowie der Ökonomie. *Transtar* ist auf Computern international renommierter Firmen wie IBM einsetzbar.
Derzeit sind Wissenschaftler der chinesischen Software Corporation maßgeblich an einem Gemeinschaftsprojekt von Experten Japans, Malaysias und Indonesiens beteiligt, dessen Ziel die Entwicklung eines automatischen Übersetzungssystems für mehrere Sprachen ist.

ADN

Vom 7. bis 18. November 1988 fand in Leipzig in 6 Ausstellungshallen des Messegeländes die 31. ZMMM der FDJ statt. Neu gegenüber dem Vorjahr war, daß sich erstmalig die Jugendverbände der RGW-Länder in einer gesonderten Ausstellung beteiligten und daß insgesamt 2651 Exponate, ein Drittel mehr als im Vorjahr, ausgestellt wurden. Fast die Hälfte davon waren Lösungen zur Mikroelektronik, CAD/CAM-Technik sowie der flexiblen automatischen Fertigung. Unter diesen Exponaten haben wir für Sie eine Auswahl getroffen, die wir Ihnen im folgenden vorstellen möchten.

31. Zentrale Messe der Meister von morgen

3

An den Beginn unseres Berichtes wollen wir die Netze stellen.

Die Ingenieurhochschule für Seefahrt Warnemünde/Wustrow stellte das Diensteintegrierte Digitale Kommunikationssystem **DDKS 1** vor. Dieses auf dem Prinzip der Puls-Code-Modulation (PCM) beruhende Vermittlungssystem vereinigt in einem Netz die Kopplung von Mikrorechnern (über V.24-Schnittstelle), Telefonen und Wechselsprechanlagen sowie die Übertragung von Fernwirk- und Fernmeßinformationen. Es ist modular so aufgebaut, daß der Preis mit der Größe des Systems linear wächst. Je 4 Teilnehmer sind zu einem Netzknoten zusammengefaßt (Bild 1); deshalb können bei gleichbleibendem Preis pro Teilnehmer zwischen 4 und 64 ... 100 Teilnehmer (netzknotenweise aufrüstbar) angeschlossen werden. Der Abstand von Netzknoten zu Netzknoten kann typisch 20 ... 60 m (maximal 1000 m!) betragen. Die Übertragung (2,304 MBit/s) erfolgt über Lichtwellenleiter. Bei Ausfall einer Übertragungsstrecke kann sich das System automatisch neu konfigurieren und notfalls auch Teilnetze bilden. Derzeit wird daran gearbeitet, durch die Realisierung der Komponenten des Systemkerns in ASICs den Aufwand für die Netzknoten drastisch zu verringern. Für das DDKS steht Kommunikationssoftware unter UDOS, CP/M 3.0, MS-DOS 3.2 und unter dem Echtzeitmultitaskbetriebssystem KOMI zur Verfügung. UNIX-Software befindet sich in Vorbereitung. Neben Treiberprogrammen wird Anwendersoftware für Filetransfer sowie Netzwerkarbeit im Hintergrund angeboten. Die Software ist kompatibel zu SCOM-LAN und LOTUNET.

(Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Sektion Schiffsführung, WB Schiffselektronik/Nachrichtendienst, Dr. Klabunde, Richard-Wagner-Straße, Warnemünde, 2530; Tel. 5 72 87)

Das Informatik-Zentrum des Hochschulwesens an der Technischen Universität Dresden war mit dem lokalen Netz **LOTUNET** vertreten. Gemeinsam mit der Friedrich-Schiller-Universität Jena, Sektion Physik/Mathematik, wurde eine Netzwerk-Interfaceeinheit (NIU) für die Rechner KC 85/2, 3, 4 sowie der Kommunikationsdienst „Verteilte virtuelle Diskette" geschaffen. Mit einer Übertragungsrate von 250 kBit/s (Bei 1,73-MHz-Takt) und einer Anschlußlänge von 500 m kann jeder Rechner auf alle im Netz vorhandenen Diskettenlaufwerke zugreifen. Dazu wurden in das Betriebssystem der KCs neue Kommandos für die Diskettenarbeit integriert. Sie befinden sich in einem EPROM auf der NIU. Für folgende Rechnertypen stehen bereits NIUs zur Verfügung: PC 1715, BC 5120/30, KC 85/1 bzw. KC 87, AC 7100/50, PC/XT bzw. AT, P 8000, SM 3/4, SM 1420, MPC 4, EC 1834, K 1840.

(TU Dresden, Informatik-Zentrum, Mommsenstr. 13, Dresden, 8027 oder FSU Jena, Sektion Physik, Max-Wien-Platz 1, Jena, 6900)

Das Institut für Informatik und Rechentechnik der AdW stellte das verteilte Briefübertragungs- und -verwaltungssystem **BVS** auf der Basis von ROLANET-Baugruppen vor (Bild 2).

Unter Nutzung der Netzkomponenten LNC 1 und TRC 1 des VEB Kombinat Robotron sind Rechner vom Typ PC 1715, A 5120, P 8000, K 8912 (modifiziert) verkoppelbar. Aus dieser Hardwarebasis sind folgende Möglichkeiten integriert: Textverarbeitung (Erstellung, Editieren), Versenden von Schriftgut sowie Archivierung und Recherche von schriftlichem Material nach bestimmten Merkmalen sowie das automatische Führen eines Postbuches.

Das System enthält Mechanismen zum Schutz vor unbefugtem Zugriff.

(AdW der DDR, Institut für Informatik und Rechentechnik, Rudower Chaussee 5, Berlin, 1199)

Aus dem Bereich der PCs und ihrer Anwendung soll zuerst der **Bildschirmarbeitsplatz für die Textbe- und Textverarbeitung** von Typoart Dresden vorgestellt werden (Bild 3).

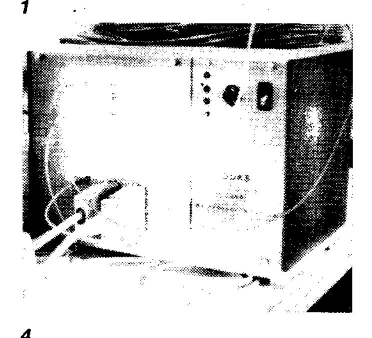
Als PC wird eine Rechnerkonfiguration auf der Basis des AC 7150 unter dem Betriebssystem Mutos 1700 eingesetzt. Mit dem angebotenen Anwendersoftwarepaket ist Vorder- und Hintergrundarbeit möglich. Es bietet unter anderem eine leistungsfähige typografische Befehlssprache, ein wählbares Maßeinheitssystem sowie eine Setzmaschinenunabhängige Auswahl von Schriften, Sonderzeichen und Sonderbuchstaben. Die Anzahl der nutzbaren Schriftfonts ist lediglich von der Größe der verfügbaren Festplatte abhängig.

Der Bildschirmarbeitsplatz kann mit der Belichtungseinheit Typoset 240 (links im Bild) gekoppelt werden. Die Typoset 240 kann Schriftgrößen von 4 ... 90 p mit einer Belichtungsbreite von 240 mm in einer Auflösung von 250, 500 oder 1000 Linien/cm ausgeben. Das System ist für die Anwendung in Druckereien (mit einer Jahresproduktion von bis zu 1000 Verlagsbogen) sowie in Redaktionen bzw. Verlagen vorgesehen.

(VEB Typoart Dresden, Großenhainer Straße 9, Dresden, 8060)

Die Sektion Informationstechnik und Theoretische Elektrotechnik der Technischen Hochschule Ilmenau zeigte das **IBS 22** (Bild 4), ein intelligentes Bildverarbeitungssystem mit Kommandosprache auf Forth-Basis. Die vorgestellte Lösung ermöglicht den Aufbau eines kostengünstigen Bildverarbeitungssystems durch Einsatz eines CP/M-Rechners mit folgenden Anwendungsgebieten:

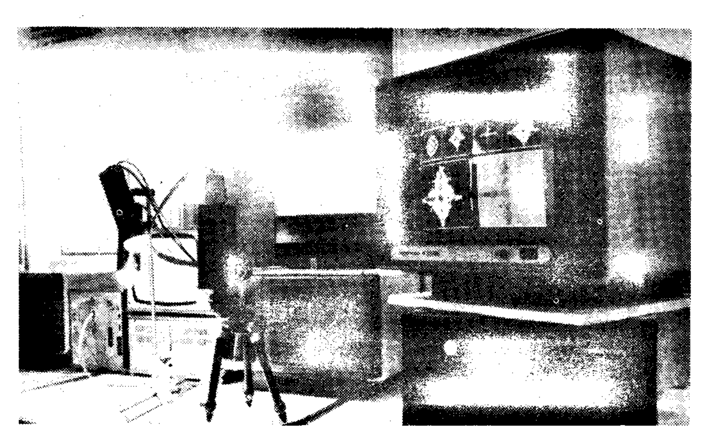
1



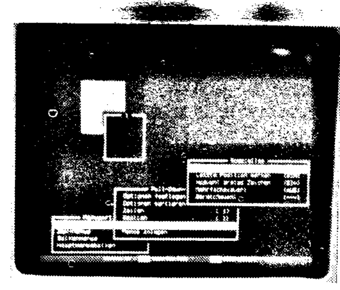
2



4

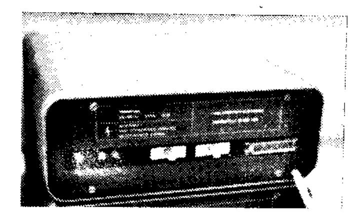


5



6

7



– **Steuerung von Industrierobotern** und Automaten auf der Basis der Objekterkennung
– **visuelle Inspektion** in Produktionsprozessen
– **interaktive Bildauswertung**, zum Beispiel in der Mikroskopie.

Es können Bilder mit einer Auflösung von 640 × 480 Pixeln mit 64 aus 4096 Farben erzeugt werden.

Die TH Ilmenau entwickelte hierfür Baugruppen zur Bilddatenaufnahme, dazu passende CCD-Kameraköpfe sowie ein Modul zur Display- und Lichtgriffelsteuerung. Die in Zusammenarbeit mit Akademieinstituten entwickelte Software steuert den Bildaufnahme- und Bilddisplayprozeß, liefert Werkzeuge zur Bilddatenstatistik, Grauwerttransformation, Bildfilterung, Merkmalsextraktion und Klassifizierung sowie zur interaktiven Bilddatenverarbeitung.

(TH Ilmenau, PSF 327, Ilmenau, 6300; Tel. 7 40)

Als Neuheit für den EC 1834 stellte das Robotron-Büromaschinenwerk Karl-Marx-Stadt eine microsoft-kompatible **Maus** (Bild 5) vor, die ab 1989 produziert werden soll. Am selben Stand wurde außerdem die neue **Version 3.30** des Betriebssystems **DCP** für den EC 1834 sowie den AC 7150 angeboten.

(VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt, Annaberger Straße 93, PF 129, Karl-Marx-Stadt, 9010)

Die Technische Hochschule Leipzig, Sektion Automatisierungsanlagen, bot ein Expertensystem für MS-DOS-fähige PCs zur Lösung höherer Automatisierungsaufgaben an. Das **Expertensystem – Shell PROCON I** dient der Unterstützung des Menschen in komplexen Entscheidungssituationen. Der hierbei realisierte Problemlösungsprozeß gliedert sich in zwei Phasen:

1. Diagnose des akuten Prozeßzustandes
2. Expertise einer resultierenden (Therapie-)Steuerung.

Damit ist es beispielsweise möglich, daß der Anlagenfahrer einer Chemieanlage in der Nachtschicht bei einer Havarie den Experten (mit aufbereitetem Expertenwissen versehener Computer) nach der Ursache, den möglichen Folgen und den in bestimmten Fristen nötigen Handlungen befragen kann.

Die bestehende Wissensbasis kann jederzeit verändert werden (Wissenseingabe). Weiterhin sind Wissensinspektion (Darstellung großer Zusammenhänge, Betrachtung von Definitionen u. a.) und Konsistenzprüfung (Kontextfreier Test auf Geschlossenheit und Konfliktfreiheit) möglich.

(TH Leipzig, Sektion Automatisierungsanlagen, Volker May, PSF 66, Leipzig, 7030)

Von der Hochschule für Ökonomie „Bruno Leuschner" Berlin war die Interessengemeinschaft „Softwaretechnologie" der Sektion Wirtschaftsinformatik auf der 31. ZMMM vertreten. Sie stellte das Window-Development-System **WDS** für Turbo-Pascal 4.0 vor. Es bietet die Möglichkeit, beliebige Fenster zu gewinnen, zu positionieren und farblich abzustimmen (Bild 6). Um den Überblick zu behalten, können fertige Fenster unsichtbar gemacht werden, so daß neue Fenster generiert werden können. Es können Menüs bei freier Wahl der Menüart und der Menüposition integriert werden. Alle Fenster- und Menükonfigurationen können abgespeichert werden, damit können später schnell Veränderungen am Oberflächen-Layout vorgenommen werden.

(I & T Software, AG Software im Klub Wissenschaft – Magnushaus, Am Kupfergraben 7, Berlin, 1080)

Eine Neuerung für PCs stellt der **V. 24/IFSS-LWL-Umsetzer** vom VEB Robotron-Projekt Dresden dar (Bild 7). Er ermöglicht die störsichere Datenübertragung zwischen mehreren Rechnern sowie zwischen Rechnern und peripheren Geräten über Lichtwellenleiter.

Voraussetzung ist die Existenz einer V. 24- oder IFSS-Schnittstelle. Denkbar ist natürlich auch die mitunter problematische V. 24-/IFSS-Umsetzung. Die Übertragungsgeschwindigkeiten betragen mit V. 24-Schnittstelle 9,6 kBaud (asynchron) bzw. 19,2 kBaud (synchron) bei einer Streckenlänge von rund 1000 m und mit IFSS-Schnittstelle 9,6 kBaud bei rund 2000 m Streckenlänge.

(VEB Robotron-Projekt Dresden, Leningrader Straße 9, Dresden, 8010)

Für viele CAD-Arbeitsplätze besteht das Problem der arbeitsgerechten Gestaltung dieser Arbeitsplätze. Die Sektion Arbeitswissenschaften der Technischen Universität Dresden hat sich dieser Aufgabe angenommen. Das Ergebnis ihrer Arbeiten ist der **Ergonomisch gestaltete CAD-Arbeitsplatz**, von dem (aus Platzgründen nur) ein Teil auf der ZMMM zu sehen war (Bild 8). Er umfaßt drei verschiedene Tische, Eckteile und einen Rollcontainer. Der *Tisch für Bildschirm und Rechner* (links im Bild) ist in zwei Ebenen (vorn und hinten) unabhängig voneinander zwischen 600 und 800 mm Höhe verstellbar. Die Teilflächen betragen je 1600 × 500 mm^2. Der *Tisch für das Digitalisiergerät* (rechts im Bild) hat eine 1000 × 700 mm^2 große Tischfläche mit einer verstellbaren Höhe zwischen 600 und 800 mm. Der nicht gezeigte *Tisch für Ausgabemedien* entspricht dem für das Digitalisiergerät, jedoch mit fester Tischplatte. Seine Höhe beträgt wie die des Eckteils 720 mm. Das Eckteil und der Rollcontainer sind in der Bildmitte zu sehen.

Bemerkenswert sind weiterhin die Softlinegestaltung der Tischkanten, die mattierte Tischfläche und die Fußauflagen.

(TU Dresden, Sektion Arbeitswissenschaften, Prof. Rentzsch, Mommsenstraße 13, Dresden, 8027)

Eine **komplexe CAD/CAM-Lösung zur Laborfertigung von Leiterplatten** stellte ein Jugendforscherkollektiv der VEB Leuna-Werke „Walter Ulbricht" vor. Mit dieser Lösung können rund 2000 durchkontaktierte Leiterplatten pro Jahr bei Losgrößen von 2 Stück gefertigt werden. Bestandteil der Lösung sind ein Digitalisierbrett, eine Zieheinrichtung, ein PC/BC oder MC 80 mit Grafikbildschirm, eine Belichtungseinrichtung, ein Farbplotter (ADMAP-4), die Laborgalvanik und die Leiterplattenbohrmaschine.

(VEB Leuna-Werke „Walter Ulbricht", Thälmannplatz, Leuna, 4220)

Das **Bestückungssystem mit Industrieroboter für gemischt bestückte Leiterplatten** (Bild 9) vom VEB Robotron-Rationalisierung Weimar ist für die vollautomatische Bestückung von konventionellen Schaltkreisen und SMDs geeignet. Dieses System ist für Klein- und Mittelserien geeignet und gewährleistet eine automatische Fehlerdiagnose und einen minimalen Umrüstungsaufwand. Im Bild ist links die Schaltkreiszuführung, in der Mitte der Industrieroboter PHM 50 und rechts die Zu- und Abführung der Leiterplatten zu sehen. Die konventionellen Schaltkreise können zur Zeit 14 bis 18 Pins haben, größere und kleinere Pinzahlen sind prinzipiell möglich. Die Größe der SMDs reicht von 1,6 × 3,2 × 0,6 mm^3 bis 13,0 × 10,6 × 2,65 mm^3.

(VEB Robotron-Rationalisierung Weimar, Hegelstraße 2a, Weimar, 5300; Tel. 37 13)

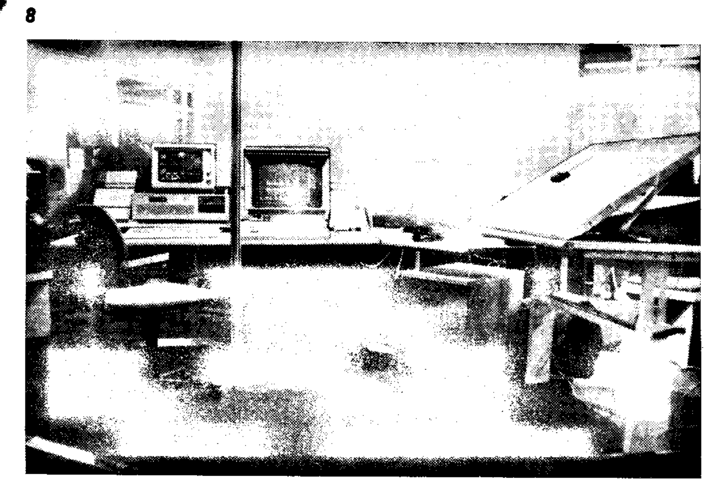
Für viele ein leidiges Problem ist die Wiederverwendung von Farbbandkassetten für Drucker. Der VEB Bekleidungswerke Erfurt bot hierfür die Lösung **Regenerierung aller Typen von Farbbandkassetten** an, bei der DDR-handelsübliche Farbbänder auf die erforderliche beliebige Breite gebracht und zu einem Endlosband verschweißt werden können.

(VEB Bekleidungswerke, Anger 33, Erfurt, 5020)

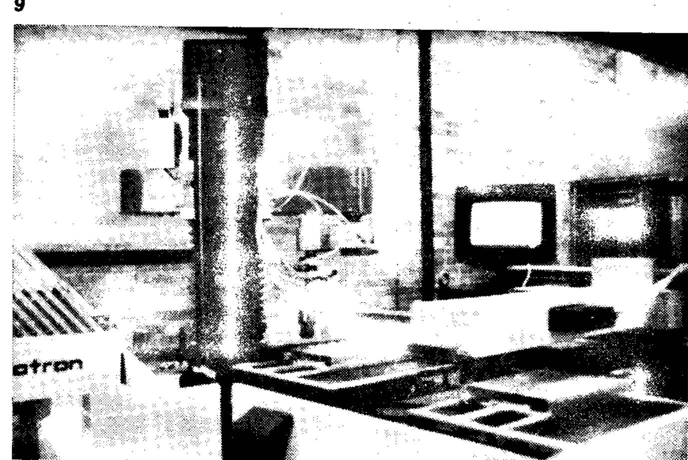
Betrachtet man die hier vorgestellten Exponate, dann fällt auf, daß einerseits einige Nachnutzungsangebote einen sehr großen Hardwareanteil besitzen (Bestückungssystem mit Industrieroboter; Ergonomisch gestalteter CAD-Arbeitsplatz) und andererseits Exponate vertreten sind, die in großen Stückzahlen in unserer Volkswirtschaft benötigt werden (DDKS 1, LOTUNET). Es wäre volkswirtschaftlich nicht sinnvoll, jedem interessierten Betrieb zu überlassen, diese Exponate in seinem Rationalisierungsmittelbau oder Musterbau nachzubauen, da viele noch offene Probleme dann hundert- und tausendfach gelöst werden müssen. Vielmehr sollte nach jeder ZMMM darüber beraten werden, welche Exponate die größten volkswirtschaftlichen Effekte bringen und entschieden werden, welcher Betrieb die Exponate in die Produktion überführt. Der Sekretär des Zentralrates der FDJ, Günter Bohn, äußerte hierzu in einem ADN-Gespräch „Nach der Messe kommt es darauf an, in der Nachnutzung eine höhere Verbindlichkeit zu erreichen, um letztlich konkrete ökonomische Effekte für den beschleunigten Leistungsanstieg zu erzielen."

Text und Fotos: Herbert Hemke

8



9



Kolloqium Datenerfassungstechnik

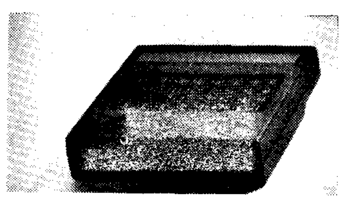


Foto: Baars

Am 4.2.1988 fand an der Technischen Universität Magdeburg ein Kolloqium zur Mobilen Datenerfassungstechnik statt. Veranstalter waren die Interessengemeinschaft Mobile Datenerfassungstechnik und die KDT-Betriebsorganisation des Zentralen Forschungsinstitutes des Verkehrswesens der DDR und der TU Magdeburg.
Eingangs wurden die Aufgaben und Ziele der IG MDET dargestellt. Es wurden die ständigen Mitglieder der IG und die durch sie repräsentierten Nutzerbereiche vorgestellt. Im Vortrag der TU Magdeburg wurde zunächst auf die Notwendigkeit und die Einordnung der mobilen Datenerfassungstechnik eingegangen. Ausgehend von den speziellen Bedingungen in Gießereien und anderen Betrieben der metallverarbeitenden Industrie wurde insbesondere auf den gegenwärtigen Stand des Einsatzes und auf Akzeptanzprobleme von Betriebsdatenerfassungsgeräten hingewiesen. Daraus wurden Anforderungen an die einzusetzende Datenerfassungstechnik, aber auch an Leitungsentscheidungen, Einsatzvorbereitung und Datenübertragungsprojektierung abgeleitet und verallgemeinert.
Vom Braunkohlenkombinat Bitterfeld wurden im Anschluß daran die in diesem Bereich in der Vergangenheit durchgeführten Entwicklungen von tragbaren Datenerfassungsgeräten und die entsprechenden Einsatzfälle vorgestellt.
Ausgangspunkt des Beitrages des ZFIV waren die Umwelt-, technischen und technologischen Anforderungen an eine solche Technik. Es wurde ausführlich auf ergonomische, arbeitsmedizinische und formgestalterische Aspekte eingegangen und ein Mustergerät mit folgenden wesentlichen Parametern vorgestellt:
LC-Matrix-Display, 8 × 4-Folienflachtastatur, 32 KByte RAM Daten- und Programmspeicher, serielle Datenübertragung über Infrarotstrahlung zu einer ortsfesten Koppelstelle, Abmessungen: 180 mm × 190 mm × 45 mm, Masse 870 g.
Das Bild zeigt das Gestaltungsmuster des Datenerfassungsterminals (Formgestaltung: VEB Designprojekt Dresden, Atelier Berlin).
Abschließend wurde kurz auf Zubehör und vorgesehene Erweiterungen sowie auf Probleme bei der Entwicklung hingewiesen. Nach einer Vorabproduktion 1988 sollen die Geräte ab 1989 in Serie durch einen Rationalisierungsmittelbetrieb der Deutschen Reichsbahn hergestellt werden.
An dieser Thematik Interessierte wenden sich bitte an das
Zentrale Forschungsinstitut für Verkehrswesen der DDR, Zentrum für Prozeßautomatisierung, Postfach 403, Berlin, 1017.

Kolb

SYSTEC 88

Vom 25. bis 28. Oktober 1988 fand in München die 2. Internationale Fachmesse für Computerintegration im Produktionsunternehmen statt. Die mit der SYSTEMS im jährlichen Wechsel stattfindende Fachmesse ist eine gelungene Ergänzung zu den etablierten Computermessen, wie z. B. der CeBit. Es steht weniger der einzelne Computer im Blickpunkt, sondern dessen Integration in ein Gesamtkonzept der vollautomatisierten „Fabrik der Zukunft". Die Herausforderung des Jahres 2000 heißt CIM (Computer Integrated Manufacturing). Sämtliche Problemkreise, die sich mit CIM-Konzepten verbinden, wurden auf der SYSTEC 88 dargestellt. Die große Bedeutung einer derartigen Fachmesse widerspiegelt die Beteiligung von 516 Ausstellern aus 12 Staaten. Auf einer Ausstellungsfläche von 35000 m^2 wurden in 16 Hallen insbesondere die Bereiche Systeme und Systemkomponenten, Entwicklung und Konstruktion, Produktion, unternehmensweite Problemlösungen, Forschung und Versuch sowie Dienstleistungen angesprochen. Doch nicht allein dieses breite Podium der Darstellung von Entwicklungsständen und -trends auf dem Gebiet CAD, CAM und CIM lockte viele Aussteller und Besucher an, insbesondere die parallel stattfindenden CIM- und CAD-Kongresse waren eine echte Bereicherung der Messe. In Plenarvorträgen und vier Parallelveranstaltungen sowie Kurztutorials und Podiumsdiskussionen war ausreichend Gelegenheit, sich mit generellen und spezifischen Problemen im CAD- und CIM-Bereich auseinanderzusetzen. Im Mittelpunkt der jeweils 2tägigen Kongresse standen die Branchen Maschinenbau, Automobil- und Fahrzeugbau, Bauwesen sowie die Verbindung zur Informatik und deren Erkenntnisse. Neben der Vorstellung firmenspezifischer Anwenderlösungen fanden besonders die Vorträge über die Schaffung von einheitlichen Referenzmodellen für CAD-Systeme, über die Normierung von CAD-Schnittstellen in CIM-Projekten große Beachtung, da sie allgemeine Entwicklungstendenzen aufzeigten.

Aufgrund der außerordentlich hohen Konzentration von Systemlösungen im Konstruktions- und Fertigungsbereich auf der SYSTEC 88 lassen sich generelle Aussagen zum Entwicklungsstand der Hard- und Softwarekonzepte zur Umsetzung der sogenannten C-Technologien ableiten.

Im Bereich der grafischen Datenverarbeitung hat sich die Rechnerklasse der Workstations weltweit durchgesetzt. Die Aussteller von Systemkomponenten für Entwicklung, Konstruktion sowie Simulation und Animation setzten zum überwiegenden Teil auf diese Hochleistungsrechner, die sowohl als Desktop-Geräte, aber auch als Tower-Modelle zu sehen waren. Als grundlegende Parameter müssen folgende genannt werden:

- RAM-Kapazität zwischen 3 und 8 MByte
- CPU-Leistung von mehr als 4 MIPS (Million Instructions per second)
- Performance in LAN (Local Area Network) von mehr als 1,2 MByte per second
- Bildschirmauflösung von mindestens 1024 × 1024 Pixels
- multitaskingfähig.

Hierbei muß festgestellt werden, daß im Bereich der grafischen Datenverarbeitung, also speziell in den Sparten CAD, CAQ, Simulation sowie der Netzsteuerung, der Übergang von der 16-Bit-Prozessorarchitektur zur 32-Bit-Prozessorarchitektur bereits vollzogen ist. Dabei setzen die Anbieter von derartigen Systemen sowohl auf die verbreiteten Prozessoren Intel 80386, Motorola 68020, National Semiconductor 32 532 als auch auf Zusatzplatinen für spezielle Anwendungen, die mit ASICs (Application Specific Integrated Circuit) bestückt sind. Ein typischer Vertreter dafür ist die Fa. Du Pont de Nemours, die ein modulares Hochleistungs-Bildverarbeitungssystem als Nachrüstplatine für die Systeme von Sun oder für den Macintosh II, die DEC-VAX und den IBM PC-AT anbieten. Einen optischen Eindruck von Hochleistungs-Workstations erhalten Sie auf der 4. Umschlagseite.

Mit der verstärkten Verbreitung der Workstations setzt sich UNIX als Betriebssystem immer mehr durch. Etwa 75% aller Softwarepakete, die im Workstation-Bereich angeboten werden, arbeiten auf der Basis von UNIX Version V.3. Aufgrund seiner Multiuser-Fähigkeit unterscheidet es sich von den PC-Betriebssystemen MS-DOS und OS/2 und macht es vergleichbar mit den Betriebssystemen der Minicomputer und Mainframes. Da es für alle Rechnerklassen verfügbar ist, wird es gerade für CIM-Projekte interessant, weil eine einheitliche Datenhaltung auf allen Levels der betriebsinternen Datenverarbeitung gewährleistet wird. Daß UNIX das Betriebssystem der Zukunft ist, wurde von Experten auf der SYSTEC auch damit begründet, daß so namhafte Unternehmen wie Nixdorf, IBM, DEC, Bull, Siemens, Philips, Apollo und Hewlett-Packard unter der Bezeichnung OSF (Open Software Foundation) bis 1990 ein standardisiertes Betriebssystem UNIX mit Echtzeitfähigkeit auf den Markt bringen wollen, das dann auch als Vorzugsbetriebssystem auf ihren Computersystemen eingesetzt werden soll.

Im Bereich der Softwarepakete für CAD-Anwendungen hat die Fa. AUTODESK mit dem Programmpaket AUTOCAD ihre marktführende Stellung auf der SYSTEC unterstrichen. Mit zahlreichen Erweiterungsmoduln für die Version 9 konnte das Anwendungsspektrum noch vergrößert werden.

Zahlreiche Firmen, wie Texas Instruments, Sun Microsystems, PSI, widmeten sich dem Bereich der künstlichen Intelligenz und ihrer Integration in CAD-Systemen. Sogenannte wissensbasierte Systeme oder Expertensysteme werden zukünftig in Form von Diagnosesystemen, Ratgebersystemen und Systemen zur präventiven Wartung einen wichtigen CIM-Baustein bilden. Die SYSTEC bot ein reichhaltiges Angebot an Software-Entwicklungswerkzeugen für wissensbasierte Systeme, im allgemeinen als Shell bezeichnet. Da Expertensysteme immer individuell aufgebaut werden müssen, werden die Shells mit konkretem Wissen der Fachleute „gefüllt".

Als Voraussetzung für die Realisierung von CIM-Konzepten gilt die vollständige informatorische Vernetzung sämtlicher Unternehmensbereiche. Erstmals in Europa wurde auf der SYSTEC ein Konzept vorgestellt, das diese Grundbedindung erfüllt und somit im CIM-Bereich Standard werden könnte. Das von General Motors Anfang der 80er Jahre in den USA entwickelte Manufacturing Automation Protocol (MAP) wurde in der Version 3.0 von der Europäischen MAP User Group (EMUG) demonstriert. Dabei wurden 20 heterogene Systeme von Firmen wie Digital Equipment, EDS, IBM, Hewlett-Packard, Motorola, Siemens, Norsk Data und Apollo über ein MAP-Netz miteinander verbunden und die Kommunikation zwischen den verschiedenen Systemen vorgeführt.

Einen großen Anteil hatte auf der SYSTEC der Bereich Qualifikation, der von CIM-Fachleuten als der wichtigste Baustein in der „Fabrik der Zukunft" bezeichnet wird. Zur Zeit entstehen völlig neue Berufsbilder, wie die des „Produktionsmechanikers" oder „Hybridfacharbeiters".

Es wird eingeschätzt, daß die Bedeutung der SYSTEC als größte CIM-Messe Europas in den nächsten Jahren noch zunehmen wird, so daß man auf die SYSTEC 90, die vom 22. bis 26. 10. 1990 wiederum in München stattfinden wird, gespannt sein darf.

Holger Benicke

DTP '88

Ende November 1988 fand im „Düsseldorfer Malkasten" eine Messe statt, die sich ausschließlich um das Thema Desktop Publishing drehte. Ins Leben gerufen wurde die DTP '88 von den beiden Düsseldorfern Manfred Leyhausen, Inhaber der Firma Leyhausen Grafikcomputer und Herausgeber der Zeitschrift Computergrafik, und Tubby Tubbax vom gleichnamigen Grafikdesignstudio. Zu den Ausstellern, die die zahlreichen Besucher über die neuesten technischen Möglichkeiten von DTP informierten, gehörte auch die Compugraphic Deutschland GmbH. Auf einem Gemeinschaftsstand präsentierten CG und die Muttergesellschaft Agfa-Gevaert AG unter anderem den Laserbelichter CG 9400 PS, den Agfa-Laserdrucker P 3400 PS und den Agfa-Scanner S 800 GS.
Besonderes Interesse beim fachkundigen Publikum fand der bereits zur Orgatechnik '88 gezeigte Laserbelichter CG 9400 PS, mit dem Compugraphic als zweiter Anbieter einer PostScript-fähigen Laserbelichter-Version nun den Markt erobern will. Mit einer Auflösung von 2400 Punkten pro Zoll (dpi) belichtet der CG 9400 PS auf Fotosatzpapier oder Film. Macintosh-Rechner und Computer des Industriestandards, die über PostScript-fähige Programme wie Ventura Publisher, Adobe Illustrator, Quark XPress und Aldus Pagemaker verfügen, können den CG 9400 PS ansteuern.

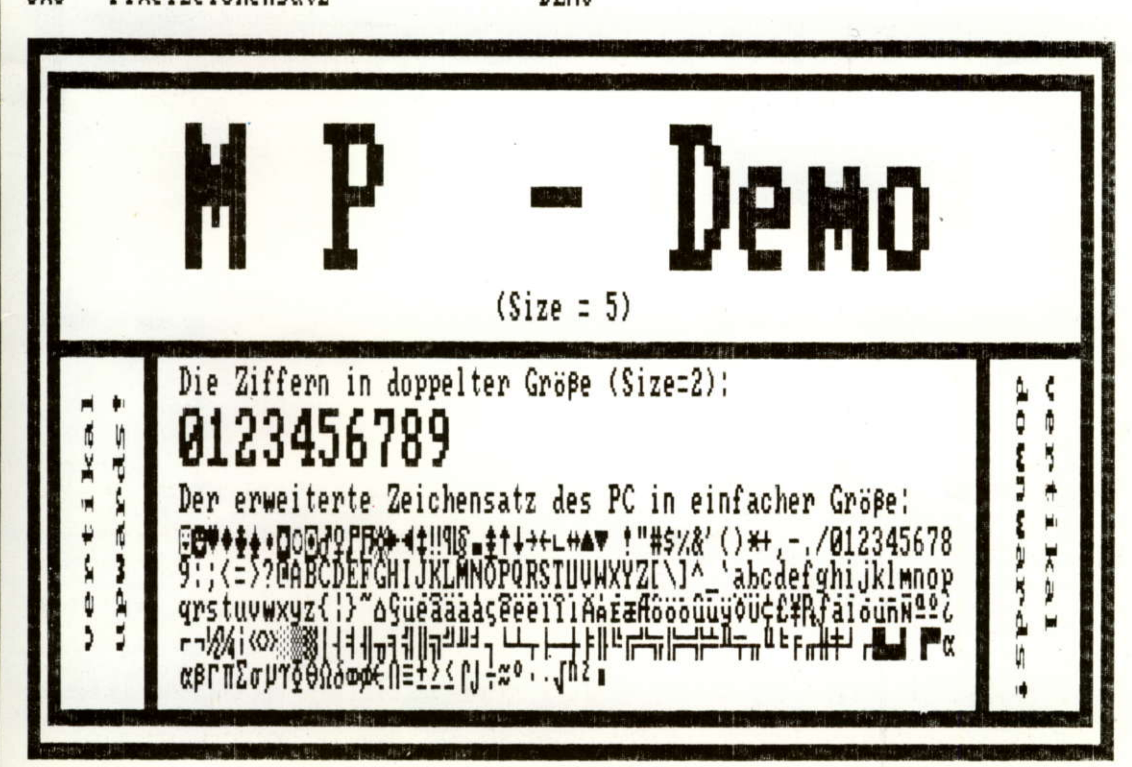
MP

.168 A8H	.169 A9H	.170 AAH	.171 ABH	.172 ACH	.173 ADH	.174 AEH	.175 AFH	.176 B0H	.177 B1H	.178 B2H	.179 B3H
.180 B4H	.181 B5H	.182 B6H	.183 B7H	.184 B8H	.185 B9H	.186 BAH	.187 BBH	.188 BCH	.189 BDH	.190 BEH	.191 BFH
.192 C0H	.193 C1H	.194 C2H	.195 C3H	.196 C4H	.197 C5H	.198 C6H	.199 C7H	200 C8H	201 C9H	202 CAH	203 CBH
204 CCH	205 CDH	206 CEH	207 CFH	208 D0H	209 D1H	210 D2H	211 D3H	212 D4H	213 D5H	214 D6H	215 D7H
216 D8H	217 D9H	218 DAH	219 DBH	220 DCH	221 DDH	222 DEH	223 DFH	224 E0H	225 E1H	226 E2H	227 E3H
228 E4H	229 E5H	230 E6H	231 E7H	232 E8H	233 E9H	234 EAH	235 EBH	236 ECH	237 EDH	238 EEH	239 EFH
240 F0H	241 F1H	242 F2H	243 F3H	244 F4H	245 F5H	246 F6H	247 F7H	248 F8H	249 F9H	250 FAH	251 FBH
252 FCH	253 FDH	254 FEH	255 FFH								

(Fortsetzung von der 2. Umschlagseite)

Lesen Sie dazu den Beitrag „8 × 8 – ein Font für alle Fälle“.

8x8 - Pixelzeichensatz DEMO



ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 2
S. 33–64

„CIM ist mehr als nur Computer"



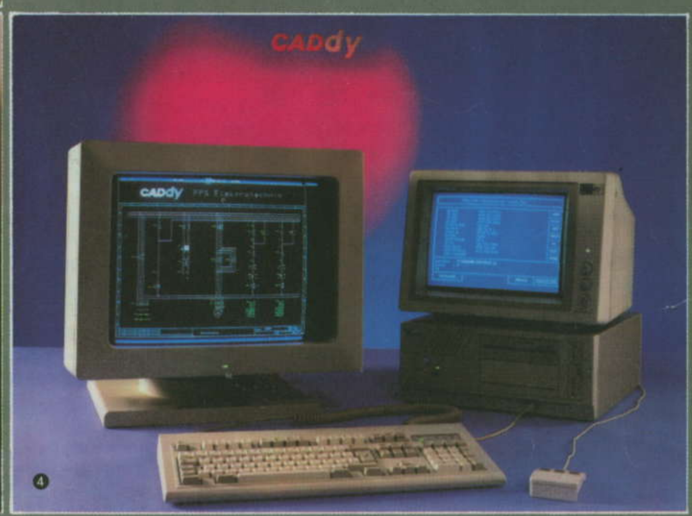
Auch das zeigte die SYSTEC 88, über die wir in diesem Heft berichten, ganz deutlich. Computer Integrated Manufacturing oder kurz CIM bezeichnet die datenmäßige Integration einer ganzen Fabrik durch den koordinierten Einsatz von Computersystemen. Erst die Summe der verschiedenen Einzeldisziplinen, wie CAD, CAM, CAQ, CAE, PPS, lassen CIM Realität werden.

Computer Aided Engineering (CAE), also das Bereitstellen von Ingenieurmethoden durch Rechnersysteme, bildet den entwicklungstechnischen Hintergrund der CIM-Strategie. CAD verkörpert das planerisch gestalterische Element von CIM. Computer Aided Design steht für das Übertragen aller Arbeiten, die bis dato mit Stift und Zeichenbrett durchgeführt werden mußten, auf den grafischen Bildschirm eines CAD-Computersystems. CAM, die rechnergestützte Produktion, ist der starke Arm von tionszeichnung zum fertigen Produkt wird durch den CAD/CAM-System-Verbund vollzogen. Daß sich die Computerintegration im Produktionsunternehmen auch qualitativ auszahlt, zeigt CAQ, das Kontrollsystem von CIM. CAQ bedeutet Qualitätsplanung und Überprüfung, frühzeitiges Erkennen von Mängeln und Qualitätsdokumentation. Damit alle Funktionen und Prozesse der Fabrik der Zukunft auch jederzeit optimal gesteuert und kontrolliert werden können, sind intelligente Produktionsplanungs- und -steuerungssysteme (PPS) im Einsatz. PPS umfaßt somit die Primärbedarfsplanung, die Materialwirtschaft, die Fertigungssteuerung, die Betriebsdatenerfassung und die Erstellung von Stücklisten und Arbeitsplänen.

Das Zusammenführen der verschiedenen Computertechniken in einer übergreifenden CIM-Strategie hat das Ziel: eine optimierte flexible und kostengünstige Produktion bei gleichzeitig verbesserter Qualität.

H. B.



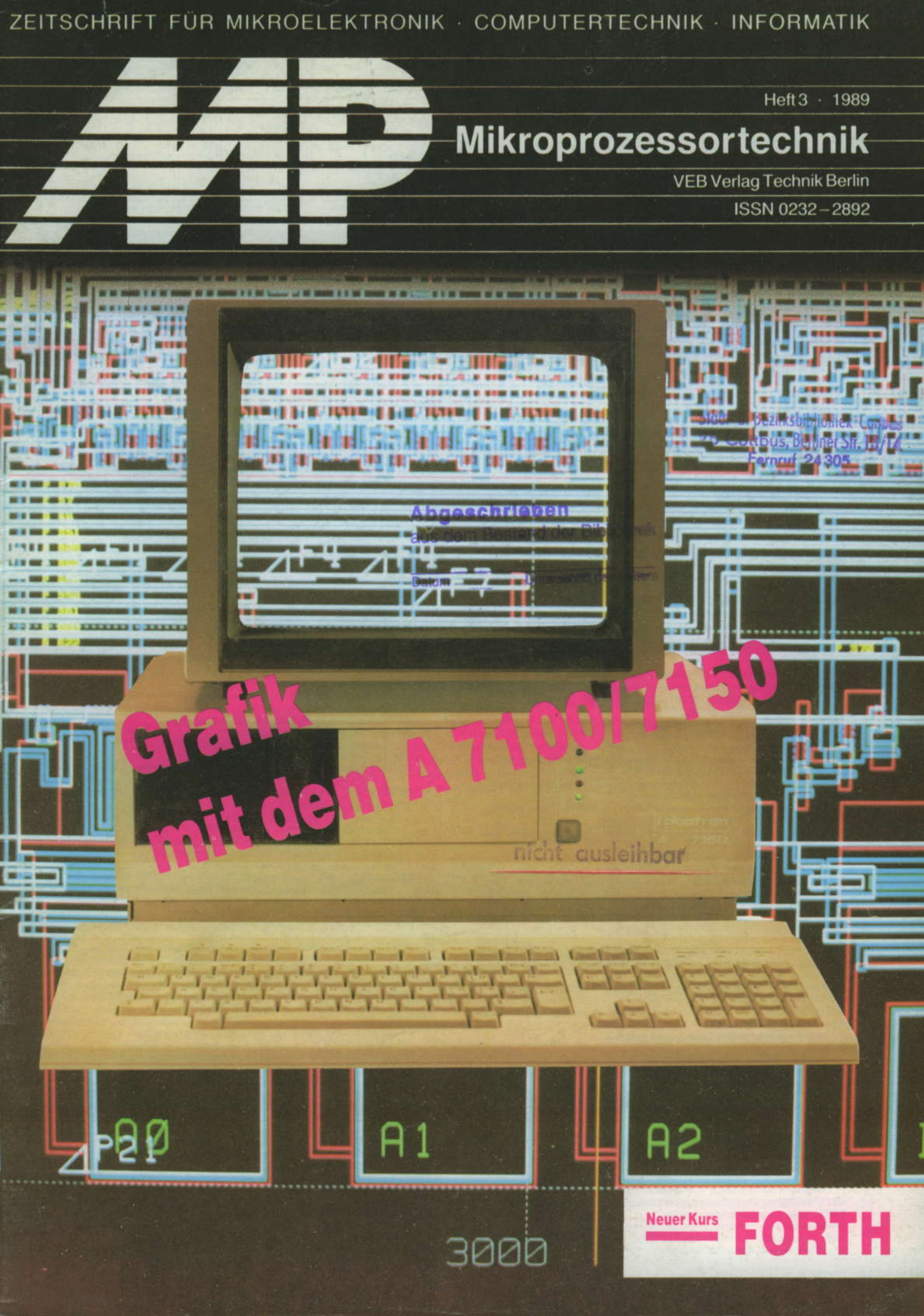


❶ ***Apollo Domain DN 3500 und DN 10000***
Neben der UNIX-Workstation DN 3500 (links), die mit Motorola-68030- und 68882-Prozessoren 4 MIPS leistet, wurde die Supercomputing-Workstation DN 10000 vorgestellt. Herzstück ist eine 64-Bit-PRISM-CPU. Bei einer Aufrüstung mit bis zu 4 RISC-CPU wird eine Arbeitsgeschwindigkeit von 100 MIPS erreicht.

❷ ***Alliant FX/40***
Von Alliant kommt ein neuer Minisupercomputer, der auf einem Universal-64-Bit-Vektorprozessor basiert. Mit 4 Hochleistungs-Vektorprozessoren werden 94,4 MFLOPS erreicht. Durch die hohe Rechenleistung und integrierte 3D-Grafik-Power eignet sich das System speziell für interaktive Bildverarbeitungsaufgaben.

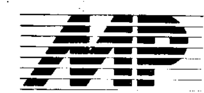
❸ ***Bravo 3***
Die CAD/CAM-Division von Schlumbergers Technologies bietet mit Bravo 3 ein modular aufgebautes System zur Entwicklung, Konstruktion und Fertigung von elektronischen und mechanischen Komponenten, Baugruppen und Gesamtsystemen. Spezielle Applikationsmodule ermöglichen die isometrische Ansicht sowie den Import von Bravo-3-Modellgeometrien. Als Hardware wird die VAX-Serie von Digital Equipment verwendet.

❹ ***CADdy von Ziegler Instruments***
Daß CAD/CAM auch auf Personalcomputern effektiv sein kann, zeigte Ziegler mit CADdy NC für die Konstruktion und Fertigung. Mit CADdy PPS steht außerdem ein effizientes Softwarepaket für das Projektmanagement in der Elektroindustrie zur Verfügung.


ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 3 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
Abgeschrieben
Fernruf 24305
Grafik
mit dem A 7100/7150
nicht ausleihbar
A0
A1
A2
3000
Neuer Kurs
FORTH

dBase III

Referenzkarte



Notation

< > Teile, die in den dBASE-Kommandos vom Nutzer zu ersetzen sind.
[ ] Teile, die wahlweise angegeben werden können.
| Alternative
{...} wahlweise Wiederholung des davor stehenden Begriffs.
SCHLÜSSELWÖRTER von dBASE-Kommandos werden in großen Buchstaben gedruckt.

Kommandos

? [<ausdr_liste>]
Übergang zur nächsten Zeile (cr, lf) und Ausgabe einer Folge von Ausdrücken auf dem Bildschirm
??<ausdr_liste>
Ausgabe einer Folge von Ausdrücken auf dem Bildschirm, ohne vorherigen Zeilenwechsel (cr, lf)
@ <zeile>,<spalte> [**SAY** <ausdr> [**PICTURE** <klausel>]]
[[**GET** <var> [**PICTURE** <klausel>]]
[**RANGE** <ausdr>[,<ausdr>]]]
@<zeile>,<spalte> **CLEAR**
Ausgabe des berechneten Wertes auf dem Bildschirm ab der angegebenen Zeile und Spalte. Anforderung eines Wertes für die Variable nach GET. <klausel> enthält Formatfunktionen und -zeichen. RANGE definiert einen Gültigkeitsbereich für numerische und Datumswerte.

ACCEPT [<prompt>] **TO** <speichervar>
Eingabe einer Zeichenkette von der Tastatur.
APPEND [BLANK] Anfügen eines Datensatzes an das Ende des aktiven Datenbankfiles.
APPEND FROM <filename> [**FOR|WHILE** <bedingung>][**SDF|DELIMITED** [**WITH BLANK** |<zeichen>]]
Anfügen eines Datenbankfiles oder eines Textfiles (SDF) an das Ende des aktiven Datenbankfiles.
ASSIST Menügesteuerte Programmierunterstützung.
AVERAGE <ausdr_liste> [<gb>] [**FOR|WHILE** <bedingung>] [**TO** <speichervar_liste>] Berechnung des arithmetischen Mittels.

BROWSE [**FIELDS** <feldliste>] Schnellkorrektur im Direktmodus (full screen mode).

CANCEL Abbruch der Programmausführung und Rückkehr zur Kommandoebene von dBASE.
CHANGE [<gb>] [**FIELDS** <feldliste>] [**FOR|WHILE** <bedingung>]
Ändern der Inhalte der angegebenen Felder.
CLEAR Löschen des Bildschirms.
CLEAR ALL Abschließen aller in Benutzung befindlichen Files (Datenbank-, Index-, Formatfiles usw.), Freigabe aller Speichervariablen und Auswahl des Arbeitsbereichs 1.
CLEAR GETS Deaktivieren der aktiven @ ... GET-Kommandos.
CLEAR MEMORY Löschen aller Speichervariablen.
CLOSE [ALTERNATE|DATABASES|FORMAT|INDEX|PROCEDURE]
Abschließen der ausgewählten Files.
CONTINUE Positionierung des Dateizeigers nach LOCATE.
COPY FILE <quell_filename> **TO** <ziel_filename>
Kopieren beliebiger Files.
COPY TO <filename> [<gb>] [**FIELDS** <feldliste>] [**FOR|WHILE** <bedingung>]
[**SDF|DELIMITED [WITH BLANK**|<zeichen>]]
Kopieren des aktiven Datenbankfiles in ein anderes Datenbank- oder Textfile.
COPY STRUCTURE EXTENDED TO <filename>
Die Struktur des aktiven Datenbankfiles wird zum Inhalt des Datenbankfiles.
COPY STRUCTURE TO <filename> [**FIELDS** <feldliste>]
Es wird ein neues Datenbankfile erstellt, mit der Struktur des aktiven Datenbankfiles und null Sätzen.
COUNT [<gb>] [**FOR|WHILE** <bedingung>] [**TO** <speichervar>]
Anzahl der Datensätze.
CREATE <filename> [**FROM** <strukturfile>] Erzeugen eines Datenbankfiles (.dbf).
CREATE LABEL <filename> Erzeugen eines Formatfiles für Briefadressen (.lbl).
CREATE REPORT <filename> Erzeugen eines Formatfiles für Bericht (.frm).

DELETE [<gb>] [**FOR|WHILE** <bedingung>]
Löschmarkierung für Datensätze setzen.
DIR [<gerät>:] [<pfad>\] [<filegruppe>] Anzeige der ausgewählten Files.
DISPLAY [<gb>] [<ausdr_liste>] [**FOR|WHILE** <bedingung>] [**OFF**] [**TO PRINT**]
Anzeige der ausgewählten Felder.
DISPLAY MEMORY [**TO PRINT**] Anzeige der gültigen Speicherplatzvariablen.
DISPLAY STATUS [**TO PRINT**] Anzeige des Status.
DISPLAY STRUCTURE [**TO PRINT**] Anzeige der Struktur des aktiven Datenbankfiles.
DO <filename> [**WITH** <parameter_liste>]
Aufruf eines Kommandofiles oder einer Prozedur.
DO CASE
CASE <bedingung> <kommandofolge>
{[**CASE** <bedingung> <kommandofolge>] ... }
[**OTHERWISE** <kommandofolge>]
ENDCASE Strukturierte Anweisung in einem Kommandofile zur Übergabe der Programmsteuerung an den ersten Fall, dessen Bedingung erfüllt ist (TRUE liefert).
DO WHILE <bedingung>
[<kommandofolge>]
[**LOOP|EXIT**]
[<kommandofolge>]

ENDDO Strukturierte Anweisung in einem Kommandofile zur zyklischen Abarbeitung der Kommandofolge, solange eine Bedingung erfüllt ist.
EDIT [[RECORD] <n>] Editieren des Datensatzes n im Direktmodus.
EJECT Seitenvorschub auf dem Drucker.
ERASE <filename> Löschen des angegebenen Files.
EXIT Unbedingtes Verlassen eines DO-Zyklus und Fortsetzung mit dem auf die DO-Anweisung folgenden Kommando.

FIND <string_ausdr> Voraussetzung: Die aktive Datenbank ist mit einem Indexfile in Benutzung. Dann bewirkt FIND die Positionierung des Dateizeigers auf den ersten Datensatz, für den Indexschlüssel und Zeichenkette gleich sind.

[**GO|GOTO**] **BOTTOM|TOP**|<ausdr> Positionieren des Dateizeigers.

HELP [<schlüsselwort>]

IF <bedingung>
<kommandofolge_1>
[**ELSE**
<kommandofolge_2>]
ENDIF
INDEX ON <ausdr> **TO** <filename> Indizieren der aktiven Datenbank nach dem angegebenen Ausdruck. Es entsteht ein Indexfile (.ndx-File).
INPUT [<prompt>] **TO** <speichervar> Sequentielle Eingabe von der Tastatur und Zuweisung des eingegebenen Wertes an die Speichervariable.
INSERT [BLANK] [BEFORE] Einfügen eines Datensatzes nach oder vor (BEFORE) dem Datensatz, auf den der Dateizeiger zeigt.

JOIN WITH <alias> **TO** <filename> **FOR** <bedingung> [**FIELDS** <feldliste>]
Vereinigen angegebener Felder von zwei Datenbanken.

LABEL FORM <filename> [**SAMPLE**] [<gb>] [**FOR|WHILE** <bedingung>] [**TO PRINT**]
[**TO FILE** <filename>] Ausgabe von Briefadressen.
LIST [OFF] [<gb>] [**FOR|WHILE** <bedingung>] [<ausdr_liste>] [**TO PRINT**]
Ausgabe der berechneten Werte der Feldliste.
LIST MEMORY [TO PRINT] Anzeige der gültigen Speichervariablen.
LIST STATUS [TO PRINT]
Anzeige des Zustandes der aktiven Datenbankfiles einschließlich Indexfiles, Alternate-Files und Systemparametern.
LIST STRUCTURE [TO PRINT] Anzeige der Struktur des aktiven Datenbankfiles.
LOCATE [<gb>] **FOR** <bedingung>
Positionieren des Dateizeigers auf den ersten Datensatz der Datenbank, der die Bedingung erfüllt.
LOOP Bewirkt innerhalb einer DO-WHILE-Anweisung die Übergabe der Programmsteuerung an den Anfang dieser DO-WHILE-Anweisung.

MODIFY COMMAND <filename> Aufruf des angeschlossenen Textprozessors zur Erstellung von Kommandofiles, i. allg. Textfiles, im Direktmodus.
MODIFY LABEL <filename>
MODIFY REPORT <filename>
MODIFY STRUCTURE Editieren eines Formatfiles oder der Struktur.

NOTE|* <text> Kommentare in einem Kommandofile mit Makroersetzung.

PACK Physisches Löschen aller Datensätze, die als gestrichen markiert sind.
PARAMETERS <parliste> Spezifizieren der Parameter einer Prozedur.
Private [ALL [LIKE|EXCEPT <gruppenbezeichnung>]] [<speichervar_liste>]
Beschränken der Sichtbarkeit von Speichervariablen.
PROCEDURE <name> Definition einer Prozedur.
PUBLIC <speichervar_liste>
Erweitern des Sichtbarkeitsbereiches der aufgeführten Speichervariablen.

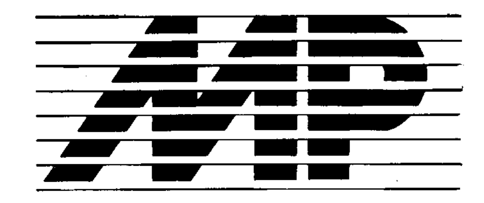
QUIT Abschließen aller eröffneten Files und Verlassen von dBASE.

READ Aktivieren des GET-Teils aller seit dem letzten READ bzw. dem Start von dBASE abgearbeiteten @ ... GET-Kommandos und Setzen des Kursors auf den Eingabebereich des ersten GET-Kommandos.
RECALL [<gb>] [**FOR|WHILE** <bedingung>]
Streichen der Löschmarkierung von Datensätzen.
REINDEX Aktualisieren aller aktiven Indexfiles.
RELEASE [<speichervar>] [ALL [LIKE|EXCEPT <gruppenbezeichnung>]]
Streichen von Speichervariablen.
RENAME <alter_filename> **TO** <neuer_filename> Umbenennen eines Files.
REPLACE [<gb>] <$feld_1$> **WITH** <$ausdr_1$>
[,<$feld_2$> **WITH** <$ausdr_2$>, ...] [FOR|**WHILE** <bedingung>]
Ersetzen von Feldinhalten.
REPORT FORM <filename> [<gb>] [**FOR** <bedingung>] [**PLAIN**] [**HEADING** <zeichenkette>] [**NOEJECT**] [**TO PRINT**] [**TO FILE** <filename>]
Ausgabe eines Berichts.
RESTORE FROM <filename> [**ADDITIVE**]
Laden von Speichervariablen vom File (.mem).
RETURN [TO MASTER] Rückgabe der Programmsteuerung an das aufrufende Kommandofile des höchsten Niveaus (TO MASTER).
RUN <kommando> Aufruf eines Kommandos aus der Umgebung von dBASE.

SAVE TO <filename> [**ALL LIKE|EXCEPT** <gruppenbezeichnung>]
Speichern aller ausgewählten Speichervariablen in ein File (.mem).
SEEK <ausdr> Positionieren des Satzzeigers auf den ersten Datensatz, dessen Index gleich dem Wert des Ausdrucks ist.
SELECT <arbeitsbereich>|<alias> Auswahl eines Arbeitsbereichs.
SET Auswahl der Optionen von dBASE im Direktmodus.
SET ALTERNATE TO [<filename>] Definition eines Alternate-Files, in welches alle sequentiell auf dem Bildschirm erscheinenden Ausgaben abgelegt werden können.
SET ALTERNATE ON|OFF An- und Abschalten der Ausgabe in das Alternate-File.
SET BELL ON|OFF An- und Abschalten eines Signals, welches bei der Eingabe des letzten Zeichens eines Feldes ertönt.
SET CARRY ON|OFF Übernahme (ON) der Daten des letzten Datensatzes in einen durch APPEND oder INSERT hinzugefügten Datensatz.
SET COLOR TO <std_display>[,<erw_display>] [<begrenzung>]
Auswahl der Anzeigearten und Farben auf dem Bildschirm.
SET CONFIRM ON|OFF Kursorsprung zum nächsten Eingabefeld.
SET CONSOLE ON|OFF Ausgabe auf den Bildschirm.
SET DEBUG ON|OFF Die Ausgabe von SET ECHO wird auf dem Drucker ausgegeben (ON).
SET DECIMALS TO <n> Festlegen der Mindestanzahl an Dezimalstellen numerischer Werte für die Ausgabe (vgl. SET FIXED).
SET DEFAULT TO <gerät> Standardgerät für den Zugriff auf Datenfiles.
SET DELETED ON|OFF Datensätze, die als gelöscht markiert sind, werden in die nachfolgenden Operationen einbezogen (OFF).
SET DELIMITER TO <zeichenkette>

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892

Grund



Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 287 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 287 03 71); Redakteure: Herbert Hemke, Hans-Joachim Hill (Tel. 287 02 03); Sekretariat Tel. 287 03 81

Gestaltung Christina Bauer

Titel Christina Bauer

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

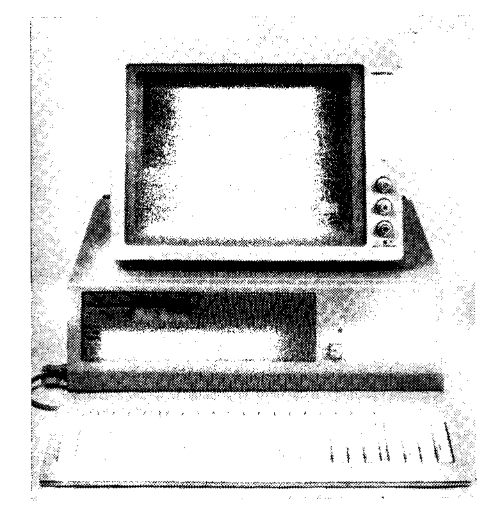
Redaktionsschluß: 10. Januar 1989

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

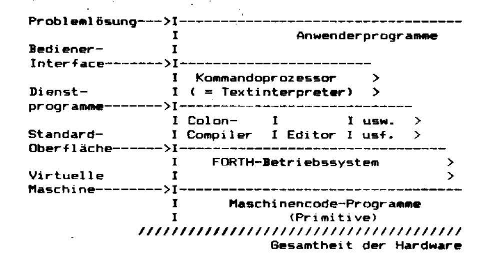
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Zum Thema Grafik mit den A 7100/7150 finden Sie auf der **Seite 67** den Beitrag „A 7100-Grafik". Er befaßt sich mit der Anwendung des SCP-GX und der KGS-Steuerfolgen in verschiedenen Programmiersprachen. Die Beiträge „Grafikbibliotheken für A 7100/7150" und „Grafikprogrammierung mit SCP-GX" auf den **Seiten 70** und **71** beschreiben eine Grafikvariante für den A 7100/A 7150 und geben Anregungen, wie die Leistungen des SCP-GX erweitert werden können.

Der dritte Teil unserer Beitragsfolge „Werkzeuge zur hardwarenahen Programmierung in höheren Programmiersprachen auf 8-Bit-Computern" auf der **Seite 75** trägt den Titel „Automatische Generierung von BASIC-INLINE-POKEs". Damit soll u. a. die Nutzung der dynamischen (Assembler-)Unterprogrammtechnik in Basic-80 erleichtert werden.



In diesem Heft beginnen wir auf der **Seite 79** mit dem Kurs „Einführung in Forth-83". Der erste Teil beinhaltet eine Einführung und die Erläuterung des Kernwortschatzes.

Vorschau
Für das Heft 4/1989 haben wir für Sie unter anderem Beiträge zu folgenden Themen vorbereitet:
- Supercomputer
- Dual-Port-RAM
- Laufzeitfehlerbehandlung
- Environment-Support für Turbo-Pascal
- dBase III Plus (mit Referenzkarte)

Inhalt

Neuer Fachausschuß „Minicomputersysteme" gebildet

Auf der Vorstandssitzung der Wissenschaftlichen Sektion (WS) *Computer- und Mikroprozessortechnik* im Fachverband Elektrotechnik der Kammer der Technik am 14. 12. 1988 wurde in Berlin ein neuer Fachausschuß (FA) für Minicomputersysteme gegründet.
Die Gründung des neuen FA zeugt von der gewachsenen Bedeutung des Einsatzes von Klein- und Mikrorechnern des Systems der Kleinrechentechnik (SKR) der sozialistischen Länder, deren Einsatz gegenwärtig alle Bereiche der Volkswirtschaft der DDR umfaßt. Ausgehend vom Einsatz der ersten 16-Bit-SKR-Anlagen vor 10 Jahren hat sich gegenwärtig der Bestand an SKR-Anlagen aus Importen und Eigenproduktion des VEB Kombinat Robotron auf über 1 000 Anlagen erhöht. Charakteristisch für die gegenwärtige Phase ist die Einführung von 32-Bit-SKR-Anlagen des Typs K 1840 und die Weiterentwicklung von 16- und 32-Bit-SKR-Anlagen zu Mikroprozessorsystemen (Elektronika-85, Elektronika MS 0104 usw.).
Ausgehend von der großen volkswirtschaftlichen Bedeutung der Minicomputertechnik besteht ein enorm hoher Bedarf am Austausch wissenschaftlich-technischer Informationen zu Einsatzvorbereitung, Nutzung, Betrieb und Weiterentwicklung der Hard- und Software der SKR-Anlagen. Bisher wurde der Erfahrungsaustausch im Rahmen der Kooperationsgemeinschaft SM3/SM4 in der DDR erfolgreich organisiert. Unter Ausschöpfung aller Möglichkeiten der Kammer der Technik stellt sich der neue FA vor allem das Ziel, in breiterem Umfang unter Berücksichtigung der Koexistenz von 16-Bit- und 32-Bit-Technik den Erfahrungsaustausch auf einem noch höheren Niveau zu organisieren. Im einzelnen stellt sich der neue FA folgende Aufgaben:
– Erfahrungsaustausch über Einsatzvorbereitung, Nutzung und Betrieb von Minicomputersystemen
– Erfahrungsaustausch über die Weiterentwicklung der Hard- und Software der Minicomputersysteme
– aktive Einflußnahme auf die Weiterentwicklung und künftige Gestaltung von Minicomputersystemen
– Organisation, Abstimmung und Koordinierung der Weiterbildungsveranstaltungen zur Minicomputertechnik
– Organisation des internationalen Erfahrungsaustausches.
Mit der Leitung der Vorstände des neuen Fachausschusses und der neuen – im folgenden genannten – Fachunterausschüsse wurde als Vorsitzender des Fachausschusses „Minicomputersysteme" Prof. Dr. sc. techn. Thomas Horn, Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden, Informatik-Rechenzentrum, beauftragt.

FUA „Architektur von Minicomputersystemen"
In diesem FUA werden generelle Fragen der Architektur von 16- und 32-Bit-Minicomputersystemen und ihrer Weiterentwicklung behandelt und ein entsprechender Erfahrungsaustausch organisiert.
Vorsitzender ist *Prof. Dr. sc. techn. Dieter Jungmann*, Informatikzentrum des Hochschulwesens an der Technischen Universität Dresden, Wissenschaftsbereich Rechnersysteme.

FUA „Systemsoftware 16-Bit-Technik"
Schwerpunkt dieses FUA ist die Organisation des Erfahrungsaustausches zur Standardsoftware für die 16-Bit-Technik, wie Betriebssysteme MOOS-1600, OS-RW, FOBOS, RAFOS, Grafiksoftware, Datenbanksysteme, Programmiersprachen und Netzsoftware.
Vorsitzender ist *Dr.-Ing. Mario Geisler*, Ingenieurhochschule Mittweida, ORZ.

FUA „Systemsoftware 32-Bit-Technik"
Schwerpunkt des FUA ist die Organisation des Erfahrungsaustausches zur Standardsoftware für die 32-Bit-Technik, wie Betriebssysteme SVP-1800, MUTOS-1800, Grafiksoftware, Datenbanksysteme, Programmiersprachen und Netzsoftware.
Vorsitzender ist *Dr.-Ing. Dietmar Gollnick*, Leitzentrum für Anwendungsforschung des VEB Kombinat Datenverarbeitung.

FUA „Applikation von Minicomputersystemen"
Auf Grund vieler Parallelen soll in diesem FUA nicht zwischen der 16- und 32-Bit-Technik unterschieden werden, obwohl in bestimmten Einsatzgebieten die 16- oder 32-Bit-Technik dominieren wird. Ziel dieses FUA ist die Organisation des Erfahrungsaustausches über die Anwendungssoftware.
Vorsitzender ist *Doz. Dr. sc. techn. Günter Bock*, Technische Hochschule Wismar, Sektion TdM.

Prof. Dr. Thomas Horn

Industriecomputer aus dem EAB

Die ersten der für 1988 geplanten 60 Industriecomputer ICA 700 aus dem VEB Elektroprojekt und Anlagenbau Berlin (EAB) wurden im Oktober 1988 an das Schwermaschinenbaukombinat „Ernst Thälmann" Magdeburg und das Werkzeugmaschinenkombinat „Fritz Heckert" Karl-Marx-Stadt ausgeliefert. Der Minister für Elektrotechnik und Elektronik, Felix Meier, sprach bei einem Besuch im Stammbetrieb des Kombinates Automatisierungsanlagenbau mit Herstellern des neuen Erzeugnisses. Jörg Bartsch, Meister im Kollektiv „Karl Marx", informierte, daß dieser ICA künftig das Haupterzeugnis des Betriebes ist. Er verwies auf die enge Zusammenarbeit zwischen den Kollegen des Musterbaus und seinem Kollektiv in der stationären Fertigung. Das helfe, so der Meister, zügig voranzukommen und 1989 400 der 16-Bit-Computer zu produzieren.
Bei der Umprofilierung des Kombinates durch erhöhten Einsatz der Mikroelektronik sollen verbesserte Arbeitsbedingungen die Werktätigen für weiteren Produktivitätsgewinn mobilisieren. Zu den neuen Anforderungen gehört die eigene Herstellung von mikroelektronischen Baugruppen und Leiterplatten in großen Stückzahlen. Ziel der Werktätigen ist es, 1990 rund 400 000 Baugruppen zu produzieren und ein Leiterplattenwerk in Betrieb zu nehmen. Etwa 600 Arbeitskräfte sind dafür nötig, die zum großen Teil durch Rationalisierung im Kombinat gewonnen werden. *ADN*

Leiterplatten für Kleinbetriebe nach Kundenwunsch

Der Ingenieurbetrieb für die Anwendung der Mikroelektronik in Erfurt entwickelt als erste Einrichtung dieser Art in der DDR auch spezielle Leiterplatten für Rationalisierungsvorhaben in Klein- und Mittelbetrieben nach Kundenwunsch. Dazu wurde ein Leiterplattenzentrum des Ingenieurbetriebes aufgebaut. Es verfügt über rechnergestützte Arbeitsplätze und entsprechende Software für den Entwurf sowie ein Labor für die Herstellung von unbestückten Prototyp-Leiterplatten. Auf der Basis des Stromlaufplans und entsprechend den Vorstellungen des Kunden entwickeln Spezialisten des Betriebes das Layout der Leiterplatte am CAD-System. Parallel dazu entstehen die Steuerlochstreifen für die Bohrautomaten. Ebenfalls am Computer werden die Schablonen für die chemische und galvanische Herstellung der Leiterzüge erarbeitet.
Seit Anfang 1988 wurde dieser Service von 18 Betrieben des Bezirkes Erfurt genutzt. Dazu zählen die Zukkerfabrik Walschleben, das Reparaturwerk „Clara Zetkin" Erfurt sowie das Energiekombinat. Jährlich können mit dieser Technologie rund 100 Aufträge bearbeitet werden. Um ihre Leistungen im Rahmen der territorialen Rationalisierung weiter zu erhöhen, sind die Mitglieder eines Jugendforscherkollektivs dabei, das Leiterplattenzentrum auszubauen.
ADN-Rothe

Robotron-Ausstellung in Budapest

Einen Überblick über sein Produktionsprofil, vor allem bei Rechen-, Meß- und Schreibtechnik, vermittelte Mitte Oktober 1988 das Kombinat Robotron mit einer Ausstellung in Budapest. Im Mittelpunkt der Exposition stand der 32-Bit-Computer K 1840, der von Robotron erstmals in Ungarn vorgestellt wurde.
Der Generaldirektor des DDR-Kombinates, Friedrich Wokurka, erklärte bei der Eröffnung unter anderem, die Ausstellung habe das Ziel, neue Möglichkeiten der Industriekooperation zwischen beiden Ländern bis hin zu gemeinsamer Produktion und gemeinsamem Absatz von Erzeugnissen zu erschließen.
Zu dem ausgestellten 32-Bit-Rechner gehörten sechs Terminals. Besondere Bedeutung besitzt er für die Industrie, beispielsweise für den Einsatz in CAD/CAM-Stationen. Vorgestellt wurden ferner Drucker sowie verschiedene Software. Wissenschaftler und Ingenieure des Kombinates sowie der Technischen Universitäten Dresden und Magdeburg erläuterten dem ungarischen Fachpublikum spezielle Programme und Möglichkeiten der Robotrontechnik.
Parallel zur eigenständigen Robotron-Ausstellung war das Kombinat auf der zur selben Zeit in Budapest stattfindenden internationalen Ausstellung Compfair 88 vertreten. Dort wurden Personalcomputer und Möglichkeiten ihres Einsatzes vorgeführt.
ADN

Gemeinsames Unternehmen für Computersysteme

Die Siemens AG sowie die Intel Corporation, Santa Clara (USA), haben vereinbart, eine gemeinsame Computerfirma zu gründen. Das Joint Venture, an dem Siemens und Intel zu jeweils 50 % beteiligt sind, erhält den Namen „BiiN" (sprich „Bein"). Eine entsprechende Vereinbarung wurde Mitte 1988 in München unterzeichnet. Das Joint Venture ergänzt die Geschäftsstrategien beider Partner und wird weltweit Computersysteme für vernetzte Rechneranwendungen besonders hoher Zuverlässigkeit anbieten. *MP*

COCOM-Restriktionen von Japan verändert

Die japanische Regierung hat im November 1988 beschlossen, die bisherigen COCOM-Restriktionen für 32 Erzeugnisgruppen zu lockern. Gleichzeitig sollen Exportbestimmungen für 31 Produkte, die für militärische Zwecke verwendet werden könnten, verschärft werden. Das Kabinett stimmte einer entsprechenden Änderung der Durchführungsbestimmungen des Gesetzes über den Devisen- und Außenhandel zu, die das japanische Ministerium für internationalen Handel und Industrie (MITI) vorgeschlagen hatte.
Aus den Embargolisten herausgenommen wurden mit Wirkung vom 20. Dezember unter anderem Computer, Transistoren, integrierte Schaltkreise und Flugzeuge. Für Minicomputer, Videorecorder und Floppy Disks ist künftig keine Exportlizenz des MITI mehr notwendig. Von COCOM-Restriktionen ebenfalls nicht mehr betroffen sind nach japanischem Außenhandelsgesetz nunmehr auch technologische Erzeugnisse wie Gallium-Arsenid-Transistoren für die Satelliten-Kommunikation.
Die japanische Regierung folgt damit einer ökonomisch motivierten Forderung einflußreicher Wirtschaftskreise Nippons, die bereits mehrfach öffentlich angeregt hatten, nur Produkte mit tatsächlicher militärischer Bedeutung für den Ost-West-Handel zu sperren. Andererseits sollen jene Verbote wegfallen, die durch technologische Fortschritte in den sozialistischen Staaten unsinnig geworden sind. Bereits im September hatte das MITI eine Liste von Hochtechnologie-Produkten – darunter elektronische Rechner, Textverarbeitungsgeräte und Frequenzmesser – veröffentlicht, deren Export erleichtert wird.
ADN

A 7100-Grafik

Dr. Klaus-Dieter Krannich, Cottbus

SCP-GX und höhere Programmiersprachen

Der A 7100 ist mit einem sehr leistungsfähigen grafischen Subsystem ausgerüstet. Mit der Betriebssystemerweiterung SCP-GX steht dem Anwender ein Werkzeug zur Verfügung, mit dessen Hilfe umfangreiche grafische Aufgaben gelöst werden können. Das Handhaben dieser grafischen Systemerweiterung als Gerätetreiber bereitet jedoch dem weniger versierten Programmierer einige Schwierigkeiten. So ist das SCP-GX von höheren Programmiersprachen, sieht man einmal von Fortran 77 ab, nicht ohne weiteres erreichbar. Selbst wenn man ein der

SUBROUTINE GDOS86 (CONTRL, INTIN, PTSIN, INTOUT, PTSOUT) (F77)

analoges Unterprogramm zur Verfügung hat, ist die Nutzung der grafischen Funktionen durch die erforderliche Belegung der Steuerfelder noch hinreichend kompliziert. Außerdem entstehen dabei recht unübersichtliche Programme, die nur durch umfangreiche Kommentare durchschaubar gemacht werden können. Es wäre deshalb wünschenswert, daß dem Anwender eine Bibliothek von Unterprogrammen zur Verfügung stände, die die jeweiligen grafischen Funktionen ausführen und denen nur die jeweils relevanten Parameter übergeben werden müssen. Die vom Hersteller für Fortran 77 gelieferten Routinen WSOPEN, WSCLCS, WSESCP, WSCHAR bilden eine gute Basis einer solchen Bibliothek. Wir wollen im weiteren eine einheitliche Realisierung dieses Konzeptes für die Programmiersprachen Turbo-Pascal, Fortran 77 und C vorstellen.

In Turbo-Pascal und C ist die Routine *gdos86* ohne große Schwierigkeiten zu realisieren. Das Parameterfeld *parms* enthält die vollständigen (je 4 Byte) Zeiger auf die Steuerfelder *contrl, intin, ptsin, intout, ptsout.* Diese Belegung erfolgt in der Routine parm_ini. In C muß man sich die Basisadresse des Datensegmentes z. B. mit Hilfe von #define DSEG (*(int *)9) aus der Basisseite beschaffen. Der Aufruf des SCP-GX erfolgt über die BDOS-Schnittstelle mit der Rufnummer 473H. Die

```
{***** Turbo-Pascal gdos86 ************************************}

type
  iptr          =^integer;
  cpuregister   = record
                  ax,bx,cx,dx,bp,si,di,ds,es,flags:integer;
                  end;

var
  contrl:array[1..35] of integer;
  intin:array[1..800] of integer;
  ptsin:array[1..800] of integer;
  intout:array[1..800] of integer;
  ptsout:array[1..800] of integer;
  parm:array[1..5] of iptr;

procedure gdos86;
var
  reg:cpuregister;
 begin
   reg.cx:=$0473;
   reg.dx:=ofs(parm);
   reg.ds:=seg(parm);
   bdos(reg);
 end;
procedure parm_ini;
begin
  parm[1]:=addr(contrl);
  parm[2]:=addr(intin);
  parm[3]:=addr(ptsin);
  parm[4]:=addr(intout);
  parm[5]:=addr(ptsout);
end;
```

Bild 1 gdos86: Turbo-Pascal

```
/***** C gdos86 ****************************

static int contrl[10],intin[800],ptsin[800],
           intout[800],ptsout[800];
static int parm[10];

#define DSEG   (*(int *)9)
#define GDOS86 (__BDOS(0x473,parm)

parm_ini()
{ int i;
  for(i=1;i<10;i+=2) parm[i]=DSEG;
  parm[0]=contrl;
  parm[2]=intin;
  parm[4]=ptsin;
  parm[6]=intout;
  parm[8]=ptsout;
}
;
; Funktion __BDOS(RUF,ARG);
;           int RUF, ARG;
;
RUF equ 4
ARG equ 6
;
cseg
___BDOS:
push bp
mov bp,sp
push di
push si
mov cx,word ptr RUF[bp]
mov dx,word ptr ARG[bp]
int 224
jmp cret
extrn cret:near
public ___BDOS
end
```

Bild 2 gdos86: C; _BDOS (ruf_nr, reg_dx)

```
;***** F77 GDOS86 ********************************************
;
;      SUBROUTINE GDOS86(CONTRL,INTIN,PTSIN,INTOUT,PTSOUT)
;      INTEGER*2 CONTROL(N),INTIN(M),PTSIN(K),INTOUT(L),PTSOUT(J)
;
;      Small Modell
;
CONTRL  equ 4
INTIN   equ 6
PTSIN   equ 8
INTOUT  equ 10
PTSOUT  equ 12
;
dseg
parm dw 0
dw seg parm
dw 0
dw seg parm
dw 0
dw seg parm
dw 0
dw seg parm
dw 0
dw seg parm
;
cseg
public gdos86
;
gdos86:
push bp
mov bp,sp
push di
push si
mov ax,word ptr CONTRL[bp]
mov parm,ax
mov ax,word ptr INTIN[bp]
mov parm+4,ax
mov ax,word ptr PTSIN[bp]
mov parm+8,ax
mov ax,word ptr INTOUT[bp]
mov parm+12,ax
mov ax,word ptr PTSOUT[bp]
mov parm+16,ax
mov dx,offset parm
mov cx,473h
int 224
pop si
pop di
mov sp,bp
pop bp
ret
end
```

Bild 3 GDOS86: Fortran 77, Small Model

```
;***** F77 GDOS86 ********************************************
;
;      SUBROUTINE GDOS86(CONTRL,INTIN,PTSIN,INTOUT,PTSOUT)
;      INTEGER*2 CONTROL(N),INTIN(M),PTSIN(K),INTOUT(L),PTSOUT(J)
;      Large-Modell
;
cseg
public gdos86
gdos86:
push bp
push ds
mov bp,sp
push bp
pop dx
add dx,8
push ss
pop ds
mov cx,473h
int 224
pop ds
pop bp
retf
end
```

Bild 4 GDOS86: Fortran 77, Large Model

Nr	Name	Parameter	Funktion
1	open_ws(nr) OPENWS(NR) open_ws(nr)	nr: Arbeitsstations-nummer	Eroeffnen einer Arbeitsstation
2	close_ws CLOSEWS close_ws()	keine	Schliessen einer Arbeitsstation
3	clear_ws CLEARWS clear_ws()	keine	Loeschen einer Arbeitsstation
4	update_ws UPDATEWS update_ws()	keine	Ausfuehren aller anstehenden Kommandos
5	set_split(nr) SETSPLIT(NR) set_split(nr)	nr: Zeilennummer	Setzen der Split-grenze
6	escape(nr) ESCAPE(NR) escape(nr)	nr: Funktionsnummer	Ausfuehren einer ESC-Funktion
7	set_lt(t) SETLT(T) set_lt(t)	t: Linientyp	Setze Linientyp
8	set_lw(w) SETLW(W) set_lw()	w: Linienstaerke	Setze Linienstaerke
9	set_lci(ci) SETLCI(CI) set_lci(ci)	ci: Farbindex	Setze Linienfarbe
10	polyline(n,feld) POLYLINE(N,FELD) polyline(n,feld)	n: Anzahl der Punkte feld: Koordinaten der Punkte x1,y1,x2 y2 ...	Zeichne Polygon
11	set_mt(t) SETMT(T) set_mt(t)	t: Markertyp	Setze Markertyp
12	set_ms(s) SETMS(S) set_ms(s)	s: Markergroesse	Setze Markergroesse
13	polymarker(n,feld) POLYMARKER(N,FELD) polymarker(n,feld)	n: Anzahl der Punkte feld: Koordinaten der Punkte	Markersymbole setzen
14	set_ch(h) SETCH(H) set_ch(h)	h: Zeichenhoehe	Einstellen der Zeichenhoehe
15	set_chup(v) SETCHUP(V) set_chup(v)	v: Neigungswinkel	Einstellen des Neigungswinkels fuer Zeichen
16	set_tc(c) SETTC(C) set_tc(c)	c: Textfarbe	Setzen der Text-farbe
17	set_tf SETTF set_tf	keine	Aktivierung der oberen Zeichen-codes
18	text(n,x,y,txt) TEXT(N,X,Y,TXT) text(n,x,y,txt)	n: Anzahl der Zeichen x,y: Punktkoordinaten txt: Text	Textausgabe
19	set_fs(s) SETFS(S) set_fs(s)	s: Fuellart	Setze Fuellart
20	set_fsi(i) SETFSI(I) set_fsi(i)	i: Fuellindex	Setze Fuellindex
21	set_fc(c) SETFC(C) set_fc(c)	c: Fuellfarbe	Setze Fuellfarbe
22	fill(feld) FILL(FELD) fill(feld)	feld: Punktkoordinaten der Ecken	Fuellen eines Polygons
23	bar(x,y,x1,y1) BAR(X,Y,X1,Y1) bar(x,y,x1,y1)	x,y: linke untere Ecke x1,y1: rechte obere Ecke	Balken zeichnen
24	circle(xc,yc,r) CIRCLE(XC,YC,R) circle(xc,yc,r)	xc,yc: Mittelpunktko-ordinaten r: Radius	Kreis zeichnen
25	arc(xc,yc,r,sw,ew) ARC(XC,YC,R,SW,EW) arc(xc,yc,r,sw,ew)	xc,yc: Mittelpunktko- r: Radius sw,ew: Start- bzw End-winkel	Kreisbogen zeichnen
26	hardcopy HARDCOPY hardcopy()	keine	
27	set_wm(m) SETWM(M) set_wm(m)	m: Schreibmodus	Setze Schreibmodus
28	set_cr(i,r,g,b) SETCR(I,R,G,B) set_cr	i: Farbindex r: Rot-Intensitaet g: Gruen-Intensitaet b: Blau-Intensitaet	Setze Farbindex

Bild 5 Liste der implementierten Funktionen

Bild 6 KGSOUT: Turbo-Pascal

```
{***** KGSOUT Turbo-Pascal **************
procedure out_kgs(c:byte);
begin
      portw[$202]:=c;
      repeat until (portw[$200] and 2)=0;
    end;
end;
```

```
;***** KGSOUT BASIC ********
;
;  Zeichenausgabe an den KGS
;
;
        cseg
;
;
OUTCHR:
        PUSH BP
        PUSH DS
        MOV BP,SP
        MOV BX,8[BP]
        MOV AL,[BX]
        MOV DX,202H
        OUT DX,AL
        MOV DX,200H
L1:     IN  AL,DX
        AND AL,2
        JNZ L1
        POP DS
        POP BP
        RETF 2
```

Bild 7 KGSOUT: Basic

```
;***** KGSOUT F77 *****
;
;  KGS - Zeichenausgabe
;
;  SUBROUTINE OUTCHR(C)
;  INTEGER*2 C
;
;  SMALL Modell
;
;
        cseg
;
        PUBLIC OUTCHR
;
OUTCHR:
        PUSH BP
        PUSH DS
        MOV BP,SP
        MOV BX,6[BP]
        MOV AL,[BX]
        MOV DX,202H
        OUT DX,AL
        MOV DX,200H
L1:     IN  AL,DX
        AND AL,2
        JNZ L1
        POP DS
        POP BP
        RET
END
```

Bild 8 KGSOUT: Fortran 77, Small Model

```
;***** KGSOUT F77 *****
;
;  KGS - Zeichenausgabe
;
;  SUBROUTINE OUTCHR(C)
;  INTEGER*2 C
;
;  LARGE Modell
;
;
        cseg
;
        PUBLIC OUTCHR
;
OUTCHR:
        PUSH BP
        PUSH DS
        MOV BP,SP
        LES BX,8[BP]
        MOV AL,ES:[BX]
        MOV DX,202H
        OUT DX,AL
        MOV DX,200H
L1:     IN  AL,DX
        AND AL,2
        JNZ L1
        POP DS
        POP BP
        RETF
END
```

Bild 9 KGSOUT: Fortran 77, Large Model

Bild 10 KGSOUT: C

```
;***** KGSOUT C *******
;
;  KGS - Zeichenausgabe
;
;  outkgs(c)
;  char c;
;
        cseg
;
        PUBLIC _outkgs
;
_outkgs:
        PUSH BP
        PUSH DS
        MOV BP,SP
        MOV AL,6[BP]
        MOV DX,202H
        OUT DX,AL
        MOV DX,200H
L1:     IN  AL,DX
        AND AL,2
        JNZ L1
        POP DS
        POP BP
        RET
END
```

Bild 11 Laden des Programmcodes in das Datensegment des Basic-Interpreters

```
60000 REM INIT-----------------
60001 OPEN "I",#1,"KGSOUT.OVL"
60002 X$=INPUT$(128,#1)
60003 DEF SEG:CLEAR ,&HFF50
60005 FOR I%=&HFF50 TO &HFF6A
60006 POKE I%,ASC(INPUT$(1,#1))
60007 NEXT
60009 CODE=&HFF50
60010 RETURN
```

Nr.	Name	Parameter	Funktion
1	init INIT init()	keine	Initialisierung
2	load_frm LOADFRM load_frm()	keine	Laden der Firmware
3	set_split(zn) SPLIT(ZN) set_split	Zeilennummer	Setzen der Splitgrenze
4	ini_screen(m,s,x1,y1,x2,y2) INISCREEN(M,S,X1,Y1,X2,Y2) ini_screen(m,s,x1,y1,x2,y2)	m: Modus s: BS-Ebene x1,y1: linke unter Ecke x2,y2: rechte obere Ecke	Initialisierung eines Bildschirmsegments
5	set_window(x1,y1,x2,y2,s) WINDOW(X1,Y1,X2,Y2,S) set_window(x1,y1,x2,y2,s)	x1,y1: linke untere Ecke x2,y2: rechte obere Ecke s: BS-Ebene	Einrichtung eines Fensters
6	set_lp(i,t,w,s) SETLP(I,T,W,S) set_lp(i,t,w,s)	i: Index (1) t: Typ (1) w: Dicke s: BS-Ebene	Setzen der Linienparameter
7	set_mp(i,t,s) SETMP(I,T,S) set_mp(i,t,s)	i: Index t: TYP (1-28) s: BS-Ebene	Setzen der Markerparameter
8	set_pen(x,y,i) SETPEN(X,Y,I) set_pen(x,y,i)	x,y: Punktkoordinaten i: Index	Anfangspunkt einer Geraden setzen
9	draw(x,y) DRAW(X,Y) draw(x,y)	x,y: Punktkoordinaten	Gerade zeichnen
10	marker(x,y) MARKER(X,Y) marker(x,y)	x,y: Punktkoordinaten (linke untere Ecke)	Marker setzen
11	set_text(x,y,t,s,txt) SETTEXT(X,Y,T,S,TXT) set_text(x,y,t,s,txt)	x,y: Punktkoordinaten t: Typ s: BS-Ebene txt: Text	Text ausgeben
12	set_arc(xm,ym,x1,y1,x2,y2,r) SETARC(XM,YM,X1,Y1,X2,Y2,R) set_arc(xm,ym,x1,y1,x2,y2,r)	xm,ym: Koordinaten des Mittelpunktes x1,y1: Koordinaten des Anfangspunktes x2,y2: Koordinaten des Endpunktes r: Radius	Ausgabe eines Kreisbogens unter Verwendung der aktuellen Linienparameter
13	set_wmode(m) SETWMODE(M) set_wmode(m)	m:Schreibmodus	Einstellen des Schreibmodus
14	select_screen(s) SELECTSCREEN(S) select_screen(s)	s: BS-Ebene	Auswahl der BS-Ebene 1 Grauwertgrafik 2 nur Ebene 1 3 nur Ebene 2 4 Ebene 1 or 2 5 Ebene 1 xor 2
15	set_fp(a,i,s) SETFP(A,I,S) set_fp(a,i,s)	a: Fuellart i: Fuellindex s: BS-Ebene	Setzen der Fuellparameter
16	fill_point(x,y) FILLPOINT(X,Y) fill_point(x,y)	x,y: Punktkoordinaten	Setzen eines Eckpunktes einer zu fuellenden Flaeche
17	fill FILL fill()	keine	Starte Fuellen
18	screen_off SCREENOFF screen_off()	keine	Dunkeltastung des Bildschirmes
19	screen_on SCREENON screen_on()	keine	Einschalten des Bildschirmes

Bild 12 Liste der Unterprogramme

```
/****************************************************************/
/* KGS - STEUERFOLGEN                                           */
/****************************************************************/
#define stx (char)2
#define etx (char)3
#define esc (char)27
char b[86];
/* KGS-Ausgabe */
kgsout(N)
int N;
{int i;
 for(i=1;i<=N;i++) outkgs(b[i]);
 return;
}
/* lo */
char lo(x)
int x;
{return((char)(x%256));
}
/* hi */
char hi(x)
int x;
{return((char)(x/256));
}
/* init */
init()
{b[1]=stx;b[2]=(char)5;b[3]=(char)0;b[4]=(char)40;b[5]=(char)1;
 b[6]=(char)1;b[7]=(char)1;b[8]=(char)1;b[9]=etx;
 kgsout(9);
}
/* set_lp (Linienparameter) */
set_lp(index,typ,width,screen)
int index,typ,width,screen;
{b[1]=stx;b[2]=(char)5;b[3]=(char)0;b[4]=(char)21;
 b[5]=lo(index);b[6]=lo(typ);b[7]=lo(width);b[8]=lo(screen);b[9]=etx;
 kgsout(9);
}
/* set_pen (Linienanfangspunkt) */
set_pen(x,y,index)
int x,y,index;
{b[1]=stx;b[2]=(char)6;b[3]=(char)0;b[4]=(char)16;
 b[5]=lo(index);b[6]=lo(x);b[7]=hi(x);b[8]=lo(y);b[9]=hi(y);b[10]=etx;
 kgsout(10);
}
/* draw (Linienendpunkt) */
draw(x,y)
int x,y;
{b[1]=stx;b[2]=(char)5;b[3]=(char)0;b[4]=(char)17;
 b[5]=lo(x);b[6]=hi(x);b[7]=lo(y);b[8]=hi(y);b[9]=etx;
 kgsout(9);
}
```

Bild 13 KGS-Steuerfolgen

Funktion _BDOS der Standardbibliothek ist nicht verwendbar, da die Rufnummer nur als Byte übergeben wird. Man ersetze den Modul BDOSS in der Standardbibliothek durch den angegebenen. Dazu erstellt man eine Datei BDOSS.A86 mit dem in Bild 2 angegebenen Quelltext, übersetzt diesen mit RASM86 BDOSS $NC und ersetzt den Modul BDOSS der Standardbibliothek mit LIB86 LIBC=LIBC.L86 REPLACE BDOSS. Die Bilder 1 und 2 zeigen die entsprechenden Quellen. Die Bilder 3 und 4 enthalten ein Assemblerlisting für eine GDOS86-Routine, die mit vom F77-Compiler (von Digital Research) erzeugten Objektmoduln verbunden werden kann. Dabei entspricht Bild 3 dem Small-Model, Bild 4 dem Large-Model. Abschließend geben wir eine Liste aller implementierten Funktionen mit einer kurzen Beschreibung der von ihnen geforderten Parameter an (Bild 5).

KGS-Steuerfolgen und höhere Programmiersprachen

Die grafische Erweiterung SCP-GX des Betriebssystems SCP 1700 bietet dem Anwender vielfältige Möglichkeiten. In vielen Anwendungsfällen werden jedoch lediglich grafische Darstellungen auf dem Bildschirm benötigt. Gerade in solchen Situationen ist die Nutzung des SCP-GX recht umständlich und eine direkte Ansteuerung des KGS über die entsprechenden Steuerfolgen wesentlich effektiver. Dann könnte die Installation des SCP-GX über das GRAPHICS-Kommando entfallen. Da auch das Laden der Grafikfirmware softwaregesteuert erfolgen kann, könnte man die Nutzerfreundlichkeit solcher Programme weiter erhöhen. Außerdem sind im SCP-GX nicht alle Möglichkeiten des KGS direkt realisiert. Hier ist vor allem die unabhängige Nutzung der beiden Bildschirmebenen interessant. Die einzige Schwierigkeit bei der Realisierung eines solchen Konzeptes besteht darin, die entsprechenden Steuerfolgen von der Ebene einer höheren Programmiersprache aus fehlerfrei an den KGS zu übergeben. Die BDOS-Funktionen zur Zeichenausgabe an die Konsole (Ruf 2 oder 6) können leider nicht benutzt werden, da diese nicht in der Lage sind, KGS-Steuerfolgen zu erkennen und gewisse Steuerzeichen vorzuverarbeiten. Damit sind natürlich auch die Mittel höherer Programmiersprachen zur Zeichenausgabe (Write-Befehle) nicht verwendbar. Es wäre möglich, aber auch recht umständlich, die entsprechenden BIOS-Funktionen (Ruf 2) zu nutzen. Die effektivste Variante besteht sicherlich darin, die Steuerfolgen direkt an den KGS auszugeben. Einen sicheren Zugriff auf die Datenports gewährleistet jedoch nur Turbo-Pascal. Bild 6 enthält eine entsprechende Prozedur. Für die Sprachen Basic 1700, Fortran 77 und C ist es erforderlich, kleine in Assembler geschriebene Unterprogramme zur Zeichenausgabe an den KGS einzubinden. Die Quelltexte sind in den Bildern 7 bis 10 zusammengefaßt. Diese Quellen sind mit RASM86 zu übersetzen, und die entstehenden Objektdateien können dann mit dem Anwenderprogramm verbunden werden. Der Basic-Interpreter erfordert eine gesonderte Behandlung. Zunächst muß der Objektmodul mit LINK86 zu einem ausführbaren Programm verbunden werden. Bild 11 zeigt, wie

man den dabei entstandenen Programmcode in das Datensegment des Basic-Interpreters ab der Adresse &HFF50 lädt. Nach der Zuweisung CODE=&HFF50 ist dann die Ausgabe des Zeichens I% an den KGS mit CALL CODE(I%) realisierbar. Damit ist es möglich, alle in der Systembeschreibung des KGS angegebenen Funktionen auszuführen. Bild 12 enthält eine Liste der von uns auf dieser Grundlage erstellten Unterprogramme mit allen erforderlichen Parametern. Zur Demonstration geben wir noch die Realisierung der Funktionen *init, set_lineparameter, set_pen* und *draw* in der Programmiersprache C an (Bild 13).

Alle so erstellten grafischen Programme lassen sich auf Quelltextebene ohne Schwierigkeiten auf den A 7150 (unter dem Betriebssystem DCP) übertragen.

Literatur

/1/ Vetter, O.: Grafik am A 7100. Mikroprozessortechnik, Berlin 1 (1987) 11, S. 325

/2/ Schmidt, E.: Hilfsroutinen zur Arbeit mit SCP-GX. Mikroprozessortechnik, Berlin 2 (1988) 4, S. 121

/3/ Dokumentation A 7100. Band 1: Rechner und Geräte, 1.56. 702001.5/53; Band 2: Logikmoduln MMS 16(MF), 1.56.702002. 3/97

/4/ Steuerprogramm SCPX 1700, Anleitung für den Systemprogrammierer.

☒ KONTAKT ☎

Ingenieurhochschule Cottbus, Karl-Marx-Straße 17, Cottbus, 7500; Tel. 692454

Grafikbibliotheken für A 7100/7150

Dr. Knut Stephan, Dietmar Thalmann
VEB Geräte- und Regler-Werke „Wilhelm Pieck" Teltow

Die 16-Bit-Arbeitsplatzcomputer A 7100/A 7150 bieten die Möglichkeit der Grafikarbeit. Dabei können unter dem Betriebssystem SCP 1700 und DCP 1700 die Grafikerweiterungen SCP-GX bzw. DCP-GX in Verbindung mit entsprechenden Grafikprozeduren genutzt werden. Für das Echtzeitbetriebssystem BOS 1810 steht eine derartige grafische Grundsoftware nicht zur Verfügung.

In diesem Beitrag wird eine Lösung vorgestellt, die es ermöglicht, unter den Betriebssystemen BOS 1810, SCP 1700 und DCP 1700 das grafische Subsystem des A 7100/A 7150 ohne Betriebssystemerweiterung zu nutzen. Diese Lösungsvariante ist für einfache grafische Anwendungen sehr zweckmäßig und effektiv einsetzbar und führt gegenüber Grafik-Betriebssystemerweiterungen zu Einsparungen hinsichtlich Rechenzeit und Speicherplatz. Die funktionelle Leistungsfähigkeit des SCP-GX/DCP-GX hinsichtlich Bedienung unterschiedlicher grafischer Geräte wird nicht erreicht.

Voraussetzungen

Grundvoraussetzung zur Arbeit mit den Grafikbibliotheken ist die Grafikfähigkeit des A 7100/A 7150, das heißt, der Arbeitsplatzcomputer muß über ein grafisches Subsystem, bestehend aus dem Controller für grafische Systeme (KGS K 7070) und der Anschlußsteuerung für grafischen Bildschirm (ABG K 7072), verfügen.

Weiterhin muß die grafische Firmware GRAFx.xxx auf Externspeicher vorliegen. Bei den Betriebssystemen SCP 1700 und DCP 1700 gehört die entsprechende Datei (z. B. GRAF6.FRM für A 7100 und GRAF7.F50 für A 7150) zum Lieferumfang des AC. Für das Betriebssystem BOS 1810 wurde diese KGS-Firmware in das entsprechende Datenformat konvertiert. Dem Laden der Firmware in den Arbeitsspeicher des KGS dient eine spezielle Prozedur der Grafikbibliothek.

Als grafisches Eingabegerät dient die Tastatur, als Ausgabegeräte der Monitor und der Drucker (HARDCOPY).

Dr.-Ing. Knut Stephan (31)
– Studium von 1977 bis 1982 an der Sektion Technische Kybernetik und Elektrotechnik der TH (jetzt TU) Magdeburg in der Fachrichtung Technische Kybernetik und Automatisierungstechnik
– von 1982 bis 1986 als wissenschaftlicher Assistent im Wissenschaftsbereich Automatisierungstechnik der Sektion Verfahrenstechnik an der TH Leuna-Merseburg
– 1987 Promotion A auf dem Gebiet der Prozeßsteuerung
– seit 1986 Mitarbeiter für F u. E im VEB Geräte- und Regler-Werke Teltow auf dem Gebiet der Automatisierungsanlagenentwicklung

Dipl.-Ing. Dietmar Thalmann (27)
– Studium von 1983 bis 1987 an der Sektion Automatisierungsanlagen der TH Leipzig in der Fachrichtung Technische Kybernetik und Automatisierungstechnik
– seit 1987 Mitarbeiter für Forschung und Entwicklung im VEB Geräte- und Regler-Werke Teltow auf dem Gebiet der Softwareerstellung für Automatisierungsanlagen

Aufbau und Wirkungsweise

Eine Grafikbibliothek ist aus einzelnen Prozeduren aufgebaut. Diese Prozeduren aktivieren die Funktionen der ladbaren KGS-Firmware des A 7100/A 7150 /1/, /2/. Dazu

a)

```
PROCEDURE OUT_KGS(DATA:FELD; M:INTEGER);
  LABEL    1;
  VAR        I : INTEGER;
      STATUS : BYTF;
BEGIN
  FOR I:=1 TO M DO
  BEGIN
  1:STATUS:=PORT[$200];                    {Einlesen KGS-Statusbyte}
    IF (STATUS AND 2)=2 THEN GOTO 1                  {Test IBF-Bit}
    ELSE
    PORTW[$202]:=LO(DATA[I]);                      {Ausgabe Daten}
  END;
END;
```

b)

```
PROCEDURE POLYLINE(ANZ:INTEGER; VAR X,Y:FELD);
  VAR  I : INTEGER;
BEGIN
  DATA[1]:=2;                                                {STX}
  DATA[2]:=6; DATA[3]:=0;    {Laenge des Funktions/Parameterteils}
  DATA[4]:=16;           {Firmwarefunktion K16: Setze Startpunkt}
  DATA[5]:=1;                         {Linienindex LI=1 --> POLYLINE}
  DATA[6]:=LO(X[1]); DATA[7]:=HI(X[1])    {Koordinaten Startpunkt}
  DATA[8]:=LO(Y[1]); DATA[9]:=HI(Y[1]);
  DATA[10]:=3;                                               {ETX}
  OUT_KGS(DATA,10);                {Datenuebergabe an KGS-Datenport}
  FOR I:=2 TO ANZ DO
  BEGIN
    DATA[1]:=2;                                              {STX}
    DATA[2]:=5; DATA[3]:=0;                               {Laenge}
    DATA[4]:=17;          {Firmwarefunktion K17: Generiere Linie}
    DATA[5]:=LO(X[I]); DATA[6]:=HI(X[I]);  {Koord. der Eckpunkte}
    DATA[7]:=LO(Y[I]); DATA[8]:=HI(Y[I]);
    DATA[9]:=3;                                              {ETX}
    OUT_KGS(DATA,9);               {Datenuebergabe an KGS-Datenport}
  END;
END;
```

Bild 1 Turbo-Pascal-Prozeduren
Bild 2 Grafik-Demonstrationsprogramm

```
PROGRAM DEMOGRAF;
{****************************************
 *    GRAFIK-DEMONSTRATIONSPROGRAMM     *
 *  TURBO-PASCAL/SCP1700/AC A7100+KGS   *
 ****************************************}

{$I PASGRAF.INC}

BEGIN
  LOAD_FIRM('A');                {Firmware von Diskette Lfw. A laden}
  INIT(2,2,3,2);                          {Anfangszustand einstellen}
  TEXT_VIDEO(3);                           {Speicherebene fuer Text}
  TEXT(235,50,16,'SCP1700 - GRAFIK');                 {Textausgabe}
  X[1]:=200; Y[1]:=40; X[2]:=400; Y[2]:=40;    {Linienkoordinaten}
  POLYLINE(2,X,Y);                                   {Linienausgabe}
  X[1]:=300; Y[1]:=200;                     {Koordinaten fuer Marker}
  POLYMARKER(1,X,Y);                                {Markerausgabe}
  CIRCLE(300,200,30);                                {Kreisausgabe}
  BAR(1,350,250,250,150);                         {Rechteckausgabe}
  FILLTYPE(3,3,2,3,0,0);                {Fuellattribute einstellen}
  X[1]:=250; Y[1]:=250; X[2]:=300; {Koordinaten der Fuellflaeche}
  Y[2]:=300; X[3]:=350; Y[3]:=250;
  FILL_AREA(3,1,X,Y);                              {Flaeche fuellen}
  LINETYPE(3,3,2);                      {Linientyp fuer Kreisbogen}
  ARC(350,200,350,250,-180);                   {Kreisbogenausgabe}
  FILLTYPE(1,3,2,8,0,0);                {Fuellattribute einstellen}
  X[1]:=250; Y[1]:=150; X[2]:=350; {Koordinaten der Fuellflaeche}
  Y[2]:=150; X[3]:=300; Y[3]:=100;
  FILL_AREA(3,0,X,Y);                              {Flaeche fuellen}
  LINETYPE(1,1,2);                       {Linietyp fuer Kreisbogen}
  ARC(250,200,250,250,135);                    {Kreisbogenausgabe}
  HARDCOPY(3);                         {Bildschirminhalt ausdrucken}
  DELAY(10000);
  CLEAR(3);                           {Loeschen des Gafikbereichs}
  SPLIT(0);                          {Alfanumerik-Teil einblenden}
END.
```

parametrierte Steueranweisungen in Form von Grafikkommandos, Spezialpuffern und Tastaturbedienkommandos mittels einer zentralen Prozedur (Bild 1a) von der ZVE direkt an den Datenport des KGS übertragen und unmittelbar nach Empfang verarbeitet. Kommt es im Rahmen der Verarbeitung einzelner Steueranweisungen zur Bereitstellung von Daten durch die KGS-Firmware (z. B. Position des grafischen Kursors, Pixelinformation), so erfolgt eine Übertragung dieser Daten vom KGS zur ZVE.

Die Prozeduren der Grafikbibliotheken werden durch Funktionsrufe (Prozeduraufrufe, CALL-Befehle) im Anwenderprogramm aktiviert. Im Bild 1b ist eine implementierte Grafikprozedur einer Turbo-Pascal-Bibliothek dargestellt (Ausgabe eines Linienzuges).

Die einzelnen Grafikbibliotheken können in Abhängigkeit vom Anwendungsfall zusammengestellt werden.

Grafikfunktionen

Die auf der Basis von Prozeduren realisierbaren Grafikfunktionen lassen sich im wesentlichen fünf Gruppen zuordnen. Zu den *Standardfunktionen* zählen das Laden der Firmware, das Einstellen der Splitgrenze und das Löschen des Grafikbereichs im Bildwiederholspeicher (BWS).

Die *Ausgabefunktionen* dienen der Darstellung von Grafikelementen (Linie, Symbol/Marker, Füllgebiet, Text, Rechteck, Kreis, Kreisbogen) auf dem Bildschirm. Außerdem ist ein Bildschirm-HARDCOPY über den ZVE-Drucker möglich. Die Koordinaten sind in den Bereichen 0..639 (X-Koordinate) und 0..399/0..479 (Y-Koordinate) anzugeben.

Mit den *Attributfunktionen* lassen sich die Attribute der Darstellungselemente (z. B. Linienstärke und -typ, Markertyp, Füllart, Helligkeit/Farbe) einstellen.

Eingabefunktionen ermöglichen die Übernahme von grafischer Information (z. B. Kursorposition, Pixelbelegung) in das Anwenderprogramm.

Steuerfunktionen realisieren im wesentlichen Parametereinstellungen (Schreibtyp, Initialisierungsmodus, Kursorart) sowie Bewegungen des grafischen Kursors.

Einbindung

Die Einbindung der Grafikbibliotheken in das Anwenderprogramm ist von der Programmumgebung abhängig.

Bei den Betriebssystemen SCP 1700/DCP 1700 muß die entsprechende Bibliothek durch $I in den Quelltext des Turbo-Pascal-Anwenderprogramms eingefügt werden (Bild 2).

Beim Echtzeitbetriebssystem BOS 1810 dagegen wird die Grafikbibliothek als Modul an das Anwenderprogramm gelinkt. Die entsprechenden EXTERNAL-Prozedurvereinbarungen sind mit $INCLUDE in die Quelle des Anwenderprogramms aufzunehmen.

Literatur

/1/ Betriebsdokumentation A 7100, Band 2, Modulbeschreibung KGS 7070 (Erweiterte Fassung). VEB Kombinat Robotron, Dresden 1986

/2/ Betriebsdokumentation A 7150, Band 2, Modulbeschreibung KGS (Erweiterte Fassung). VEB Kombinat Robotron, Dresden 1987

☒ KONTAKT ☏

VEB Geräte- und Regler-Werke Teltow, Abteilung TSA, Oderstraße 74–76, Teltow, 1530; Tel. 442245

Grafikprogrammierung mit SCP-GX

Prof. Dr. Manfred Vogel, Andreas Eger
Techn. Universität Karl-Marx-Stadt

Die Bildschirmauflösung der Arbeitsplatzcomputer A 1700 und A 1750 von 640 × 400 Bildpunkten bei einer Darstellungsfläche von 200 × 137,6 mm² gestattet die Ausgabe von Layouts, Plänen, Funktionsverläufen, Modellen und anderen ingenieurtechnischen Darstellungen. Die Erzeugung von Abbildungen von Körpern variabler Abmessung und Gestalt erfordert die Möglichkeit bequemer Transformationen zwischen verschiedenen Koordinatensystemen. Im Unterschied zum grafischen Kernsystem GKS unterstützt die grafische Betriebssystemerweiterung SCP-GX nur die Transformation auf die Gerätekoordinaten (DC) des gewählten physischen Ausgabe- bzw. Eingabegerätes. Das heißt, daß beim Aufruf einer SCP-GX-Funktion die Koordinaten sämtlicher Punkte in normalisierten Gerätekoordinaten (NDC) mit Werten im Bereich von 0 bis 32767 angegeben werden müssen bzw. zurückvermittelt werden.

Der Ursprung des NDC-Koordinatensystems liegt in der linken unteren Ecke der Darstellungsfläche, die normalisierten Gerätekoordinaten werden auf jede Achse des E/A-Gerätes komplett abgebildet.

Für die praktische Arbeit des Anwenderprogrammierers verbinden sich damit folgende Probleme:

- Der Ingenieur ist es gewohnt, für zeichnerische Darstellungen im allgemeinen in mm zu denken.
- Er beschreibt seine Darstellung in einem für den jeweiligen Zweck geeigneten Koordinatensystem, dem Weltkoordinatensystem (WC). Dieses Weltkoordinatensystem ist in seinen Achsen entsprechend den benötigten physikalischen Einheiten geteilt. Sein Ursprung liegt, z. B. aus Symmetriegründen, nicht unbedingt in der linken unteren Ecke.
- Bei vielen Aufgabenstellungen, vor allem aus dem CAD-Bereich, wird eine maßstäbliche, unverzerrte Abbildung geometrischer Objekte gefordert. Voraussetzung dafür ist eine isotrope Skalierung, das heißt die Abbildung des Weltkoordinatensystems im gleichen Maßstab auf beide Achsen des Gerätekoordinatensystems (DC).
- Die Darstellungsflächen der unterschiedlichen E/A-Geräte (z. B. Bildschirm, Plotter, Drucker) unterscheiden sich sowohl in der absoluten Größe als auch im Seitenverhältnis. Bedingt durch die SCP-GX-Konvention, daß die NDC-Koordinaten von 0 bis 32767 auf jede Achse des E/A-Gerätes abgebildet werden, ist somit für jede Achse jedes verwendeten Gerätes eine unterschiedliche Transformation notwendig.

Bild 1 veranschaulicht die unterschiedlichen Transformationen. Ein Blechteil sei im Weltkoordinatensystem x,y beschrieben. Wesentlich für das Finden einer Transformationsvorschrift ist der abzubildende Bereich des Weltkoordinatensystems, der durch die Punkte P1(XA,YA) und P2(XE,YE) beschrieben wird (Fenster). Dieses Teil soll auf Bild-

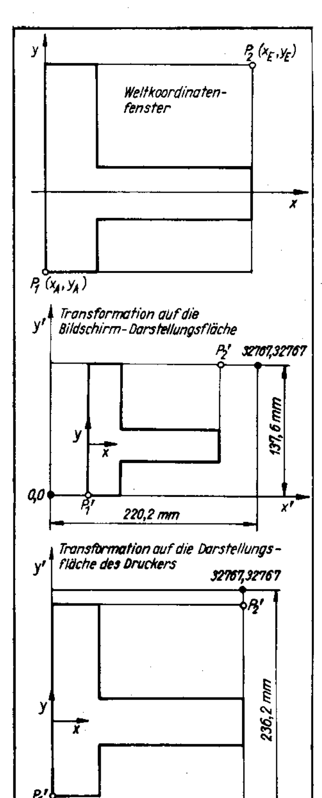


***Bild 1** Darstellung eines geometrischen Objektes auf unterschiedlichen Grafikgeräten mit isotroper Skalierung*

***Bild 2** Unterprogramme ISO und ANISO*

```
SUBROUTINE ISO
COMMON /TRANS/ IUA,IVA,XA,YA,XE,YE,RPXM,RPYM
COMMON /GRAF/ WSN,CONTRL,INTIN,PTSIN,INTOUT,PTSOUT
INTEGER*2 WSN,CONTRL(10),INTIN(80),PTSIN(400),
INTOUT(45),PTSOUT(12)
B1=REAL(INTOUT(1))
B2=REAL(INTOUT(2))
B4=REAL(INTOUT(4))
B5=REAL(INTOUT(5))
B=B1*B4
H=B2*B5
RPX=32767/B
RPY=32767/H
RM=B/H
BA=B
HA=H
DX=XE-XA
DY=YE-YA
R=DX/DY
IF (R.GT.RM) HA=B/R
IF (R.LT.RM) BA=H*R
IUA=INT(RPX*(B-BA)/2)
IVA=INT(RPY*(H-HA)/2)
RPXM=RPX*BA/DX
RPYM=RPY*HA/DY
RETURN
END

SUBROUTINE ANISO
COMMON /TRANS/ IUA,IVA,XA,YA,XE,YE,RPXM,RPYM
DX=XE-XA
DY=YE-YA
IUA=0
IVA=0
RPXM=32767./DX
RPYM=32767./DY
RETURN
END
```

schirm und Drucker K 6313 unverzerrt ausgegeben werden, wobei die jeweils zur Verfügung stehende Darstellungsfläche optimal ausgenutzt werden soll. Da die Seitenverhältnisse der Ausgabegeräte nicht dem Seitenverhältnis des Weltkoordinatenfensters entsprechen, bleibt zwangsläufig ein Teil der Darstellungsfläche ungenutzt. Die Abbildung des Weltkoordinatenfensters soll jedoch in der Mitte der Darstellungsfläche liegen.
Zur Realisierung der Transformation sind für jede Achse zwei Werte zu bestimmen:
- der Wert, der die Lage des linken unteren Punktes der Abbildung des Weltkoordinatenfensters in NDC angibt (im Programm IUA bzw. IVA)
- ein Maßstabsfaktor für die Abbildung der Weltkoordinaten auf NDC (im Programm RPXM bzw. RPYM).

Diese Werte werden für die angegebene Größe des Fensters und das vorgesehene E/A-Gerät im Programmablauf einmalig bestimmt. Für isotrope Skalierung geschieht dies im (Fortran 77-) Unterprogramm ISO, für anisotrope Skalierung (unterschiedliche Achsteilung), wird das Unterprogramm ANISO bereitgestellt (Bild 2). ISO errechnet die Größe der physischen Darstellungsfläche des jeweiligen E/A-Gerätes aus den beim Initialisieren der Arbeitsstation im Ausgabeparameterbereich INTOUT vermittelten Werten. Für die Vermittlung der Transformationsparameter wurde im Beispiel der benannte COMMON-Block /TRANS/ eingeführt. Er enthält acht Werte mit folgender Bedeutung:

IUA, IVA INTEGER*2) Lage des linken unteren Punktes der Abbildung des Weltkoordinatenfensters in NDC
XA, YA, XE, YE (REAL*4) linker unterer und rechter oberer Punkt des Weltkoordinatenfensters in WC
RPXM, RPYM (REAL*4) Umrechnungsfaktor WC in NDC in Richtung der x- bzw. Y-Achse

Mit Hilfe der im COMMON /TRANS/ bereitgestellten Transformationsparameter erfolgt die Transformation eines Punktes von WC in NDC über die Funktion IX bzw. IY (Bild 3). Für die Rücktransformation von normalisierten Gerätekoordinaten in Weltkoordinaten, z. B. bei der LOCATOR-Eingabe, stehen die Funktionen RX bzw. RY zur Verfügung. Die Belegung des Eingabepunktbereiches PTSIN erfolgt damit nicht mehr direkt, sondern über die genannten Funktionen:
z. B.: PTSIN(1)=IX(X1)
PTSIN(2)=IY(Y1).
X1 und Y1 sind die Koordinaten eines Punktes des Weltkoordinatensystems innerhalb des angegebenen Fensters, PTSIN enthält die normalisierten Gerätekoordinaten.

```
INTEGER*2 FUNCTION IX(X)
COMMON /TRANS/ IUA,IVA,XA,A,XE,B,R,D
IF (X.GT.XE) THEN
IX=32767
ELSE IF (X.LT.XA) THEN
IX=0
ELSE
IX=IUA+INT(R*(X-XA))
ENDIF
RETURN
END

INTEGER*2 FUNCTION IY(Y)
COMMON /TRANS/ IUA,IVA,A,YA,B,YE,D,R
IF (Y.GT.YE) THEN
IY=32767
ELSE IF (Y.LT.YA) THEN
IY=0
ELSE
IY=IVA+INT(R*(Y-YA))
ENDIF
RETURN
END
```

Bild 3 Funktionen IX und IY

☒ KONTAKT ☏
Technische Universität Karl-Marx-Stadt, Sektion Maschinen-Bauelemente, Straße der Nationen 62, Karl-Marx-Stadt, 9010

Erfahrungen beim Übergang von dBase II zu dBase III

Dr. Wilfried Grafik[1]
Humboldt-Universität zu Berlin, Sektion Mathematik

Die Datenbanksysteme dBase II und dBase III gehören in der 8- und 16-Bit-Technik, neben der Textverarbeitung und der Tabellenkalkulation, zu den am meisten benutzten Programmen. Viele Datenbank-, Kommando-, Format- und Berichtfiles wurden unter dBase II erstellt. Beim Umstieg auf dBase III, aber auch bei notwendiger gleichzeitiger Nutzung von 8- und 16-Bit-Technik, stehen deshalb zwei Fragen im Vordergrund: Wie können die unter dBase II vorhandenen Files unter dBase III genutzt werden? Welche Unterschiede bestehen zwischen dBase II und dBase III? Die letzte Frage läßt sich leider nicht durch einen Überblick oder eine Tabelle beantworten. Alle diesbezüglichen Versuche sind zwangsläufig unvollständig und liefern für wesentliche Details keine befriedigenden Antworten. Es bedarf dazu einer eingehenden Beschäftigung mit dBase III. Deshalb sei hier auf ein Buch verwiesen /1/.
Der Frage der Konvertierung von dBase II nach dBase III und den dabei auftretenden Problemen wollen wir uns in diesem Artikel zuwenden. Verbreitet ist das Konvertierprogramm *dCONVERT* zum Konvertieren aller in dBase II benutzten Filetypen. Die Untersuchungen beziehen sich auf die Version 1.07, das heißt, bei höheren Versionen oder kompatiblen Programmen können einige der hier auftretenden Anomalien beseitigt sein. Eine Auflistung der bekannten Probleme, gegebenenfalls ihre Überprüfung, dürfte in jedem Fall wertvoll sein.
dConvert ist ein Programm für die unvollständige Konvertierung von dBase II-Files. Das entspricht dem allgemeinen Trend, mit Programmen eine Hilfe anzubieten, selbst wenn diese nicht alle Anwenderprobleme überdeckt. Es ist also kein Fehler des Programms, wenn einige generierte Files nachträglich vom Nutzer zu überprüfen sind. Tatsächlich werden die meisten Files in ihrer Semantik völlig korrekt überführt. Dessenungeachtet müssen wir uns hier auf die derzeit bekannten Probleme konzentrieren. Dazu untersuchen wir die Konvertierung der einzelnen Filetypen.
Beim Transport aus dem Betriebssystem CP/M dürfen die Filenamen keine Sonderzeichen enthalten! Kommandofiles mit der Namenserweiterung *.cmd* müssen in die Erweiterung *.prg* umbenannt werden.

Datenbankfiles

Eine Konvertierung der *.dbf*-Files ist erforderlich, da die internen Strukturen in dBase II und dBase III unterschiedlich sind. Enthalten die Daten des Datenbankfiles keine Umlaute oder andere Sonderzeichen und wird für logische Konstanten immer T, t, F oder f verwendet, so ist die Konvertierung korrekt. Werden **Umlaute** verwendet, so ist deren Konvertierung von ihrer Darstellung in dBase II abhängig. In jedem Fall erfolgt keine Konvertierung des internen Kodes dieser Zeichen. Ihre Darstellung in dBase III ist damit meist nicht korrekt. So benutzen einige 8-Bit-Versionen die Kodes der Zeichen { | } [\ ] ~, also 7B, 7C, 7D, 5B, 5C, 5D, 7E, zur Darstellung der Zeichen ä, ö, ü, Ä, Ö, Ü, ß. Das entspricht den möglichen nationalen Abwandlungen des 7-Bit-ASCII-Kodes. In dBase III erscheinen jedoch die zuerst aufgeführten Zeichen.
Ferner existieren 16-Bit-Versionen von dBase II, die eine programminterne Darstellung dieser Zeichen verwenden. ä, ö, ü, Ä, Ö, Ü, ß haben dort die internen Kodes D4, E4, D1, DE, E9, EA, F9, was in dBase III den Zeichen von Satz 3 in Bild 1 entspricht. Ist eine nachträgliche Konvertierung im Direktmodus zu aufwendig, so bieten sich zwei Wege an. In jedem Fall ist ein eigenes Konvertierungsprogramm erforderlich, welches die internen Kodes umwandelt. Da es in Turbo-Pascal oder C leicht zu erstellen ist, verzichten wir hier auf die Angabe. Vielmehr steht die Frage, auf welches File dieses Programm anzuwenden ist? Auf ein dBase II- oder ein dBase III-Datenbankfile (.dbf)? In beiden Fällen darf der Kopf des Datenbankfiles nicht konvertiert werden. Da der Kopf eines dBase III-Datenbankfiles von der Anzahl der Felder im Datenbankfile abhängt, muß der Datenanfang erst ermittelt werden. Er errechnet sich aus

$$32 + n * 32 + 2,$$

wobei n die Anzahl der Felder im Datenbankfile ist. Der interne Aufbau eines dBase III-Datenbankfiles ist in /1/ dokumentiert. In dBase II hat der Kopf eines Datenbankfiles die konstante Länge von 512 + 8 Byte, das heißt, das erste Datenbyte, und damit das erste zu konvertierende Byte, hat vom Fileanfang die Verschiebung 520. Ein weiterer Ausgangspunkt für die Konvertierung der Umlaute im Datenbankfile kann ein Textfile sein. Es kann ohne die Kenntnis interner Dateistrukturen vom ersten bis zum letzten Byte konvertiert werden. Es wird in dBase II erzeugt durch

COPY TO ⟨file⟩ SDF

Das Einlesen als dBase III-Datenbankfile erfolgt mit

APPEND FROM ⟨file⟩ SDF,

wobei zuvor ein leeres Datenbankfile mit der durch das Textfile gegebenen Struktur erzeugt und benutzt wird (CREATE, USE).
Umlaute und Sonderzeichen sind offenbar

1) Teile der hier dargelegten Untersuchungen wurden in einer studentischen Arbeit von Thomas Ganschow und Astrid Schulz erbracht.

```
. USE mp1
* Durch das Konvertierungsprogramm erzeugtes File
. LIST

Satz #  ARTIKEL          PREIS AM_LAGER GELIEFERT BEM
    1   Butter            2.40 .T.         12.02.89  Sorte 1
    2   Brot              1.05 .T.         08.03.88
    3   ʟΣ╤▐θΩ·           0.00 .F.         29.02.89  Test Datum/Umlaute
. MODIFY STRUCTURE
   .... ändern des Typs des Feldes 'geliefert' in 'Datum'
   3 Sätze addiert
* Nach der Typänderung wurde auch der 29.2.89 in den 1.3.89
* umgewandelt
. LIST

Satz #  ARTIKEL          PREIS AM_LAGER GELIEFERT BEM
    1   Butter            2.40 .T.         12.02.89  Sorte 1
    2   Brot              1.05 .T.         08.03.88
    3   ʟΣ╤▐θΩ·           0.00 .F.         01.03.89  Test Datum/Umlaute
```

Bild 1 Typkonvertierung und Darstellung der Umlaute

```
. DISPLAY STRUCTURE

Strukturdaten für Datei: C:MP1      .DBF
Anzahl der Sätze:   00003
Datum der letzten Aktualisierung: 11/13/88
Primäre Datei
Feld        Name         Typ  Länge    Dez.st.
001        ARTIKEL        C    010
002        PREIS          N    .008      002
003        AM:LAGER       L    001
004        GELIEFERT      C    008
005        BEM            C    020
** Gesamt **                   00048

. LIST

00001  Butter               2.40 .T. 12.02.89 Sorte 1
00002  Brot                 1.05 .T. 08.03.88
00003  äöüÄÖÜß              0.00 .F. 29.02.89 Test Datum/Umlaute
```

Bild 2 dBase II-Datenbankfile mp1.dbf

bei Feldern vom Typ *character* (Zeichen) von Bedeutung. Felder vom Typ *numeric* werden ohne Beanstandungen konvertiert. Felder vom Typ *logical* werden dann falsch konvertiert, wenn als Feldinhalt in einer deutschsprachigen dBase II-Version *J* oder *j* vorkommt. Dieser Wert wird in den undefinierten logischen Wert *?* konvertiert. Er wird sichtbar, wenn der betreffende Satz über EDIT oder CHANGE betrachtet wird. Als logischer Wert wird *F* für FALSE geliefert. *Y* und *y* werden dagegen semantisch korrekt in *T* (TRUE) überführt. *T*, *t*, *F* und *f* werden übernommen, was ein weiterer Grund dafür ist, bei der Datenerfassung nur diese als logische Konstanten zu verwenden.

Ist der Übergang zu den in dBase III neuen Typen *date* (Datum) und *memo* möglich? Das Konvertierungsprogramm berücksichtigt derartige Typtransformationen nicht. Eine Lösung kann deshalb nur in dBase III gesucht werden. Für den Fall der **Memo-Felder** ist die Antwort schnell gegeben: Es gibt keine Möglichkeit, Memo-Feldern im sequentiellen Modus Inhalte zuzuweisen, das heißt, die Konvertierung ist nicht möglich.

Zur Umwandlung einer Zeichenkette in den **Typ Datum** betrachten wir das dBase II-Datenbankfile *mpf.dbf* (16-Bit-Version 2.41), dessen Struktur und Inhalt im Bild 2 wiedergegeben ist. Der letzte Satz des Files soll die Darstellung der Sonderzeichen veranschaulichen. Nach der Konvertierung erhalten wir den im Bild 1 im oberen Teil dargestellten Inhalt. Über MODIFY STRUCTURE wird der Typ des Feldes *geliefert* in *Datum* geändert. Es erfolgt eine korrekte Konvertierung der Datumswerte (Zeichen) in Werte des Typs *Datum* von dBase III. Die Einhaltung der Datumsgrenzen für die Tage und Monate wird überprüft.

Aus Bild 1 ist ferner ersichtlich, daß der **Feldname** *am:lager* korrekt in *am_lager* verändert wurde. Probleme entstehen, wenn der Transport einer Feldbezeichnung in einem Datenbankfile erfolgt. Solche Datenbankfiles können mit COPY STRUCTURE EXTENDED erstellt werden (Bild 3). Für das Konvertierungsprogramm sind diese Datenbankfiles nicht von anderen unterscheidbar, das heißt, der Doppelpunkt in Feldnamen wird nicht konvertiert (Bild 4). Soll dieses File dann in dBase III zur Definition eines Datenbankfiles benutzt werden (CREATE FROM ⟨file⟩), so wird ein falscher Feldname erkannt.

Kommandofiles

Die Konvertierung der Kommandofiles bereitet die meisten Probleme. Das ist dadurch bedingt, daß sich nicht nur die Bezeichnungen einiger Funktionen verändert haben, sondern daß auch semantische Änderungen in den Kommandos vorgenommen wurden. Nach der Konvertierung jedes Kommandofiles erscheint eine Mitteilung, die auf mögliche Fehler hinweist. Nachweisbar können aber auch dann Fehler im erzeugten Kommandofile enthalten sein, wenn eine einhundertprozentig erfolgreiche Konvertierung mitgeteilt wird. Betrachten wir die Unterschiede im einzelnen.

Wird eine Zeichenkette über **ACCEPT** angefordert und als Eingabe die ENTER-Taste betätigt, so liefert sie ein Leerzeichen in dBase II, die leere Zeichenkette in dBase III:

```
* dBASE II: Ein ':' wird selbständig an die An-
* forderungszeichenkette (prompt) angefügt
. ACCEPT ":" TO m
::              ((es wurde ENTER eingegeben))
. DISPLAY MEMORY
M               (C)
** Gesamt ** 01 Variablen benutzt
             0002 Bytes belegt
. ? LEN(m)
  1

* ... und in dBASE III
. ACCEPT ":" TO m
:
. DISPLAY MEMORY
M                 lokal (priv) C ""
   1 Variable definiert        2 Bytes benutzt
 255 Variable verfügbar,    5998 Bytes verfügbar
```

LOCATE liefert beim Gültigkeitsbereich ALL und erfolgloser Suche in dBase II als aktueller Satznummer (#) die Nummer des logisch letzten Satzes.

```
. USE mp1
. LOCATE FOR artikel="Milch"
Dateiende erreicht
. ? #, EOF()
          3 .T.
. ? artikel,preis, am:lager, geliefert
äöüÄÖÜß         0.00 .F. 29.02.89
```

In dBase III liefert die gleiche Anweisung die physisch höchste Satznummer plus 1 und beim Zugriff auf die Felder die zum jeweiligen Typ gehörigen Initialisierungswerte.

```
. USE mp1
. LOCATE FOR artikel="Milch"
Ende des LOCATE-Bereiches
. ? RECNO(), EOF()
          4 .T.
. ? artikel, preis, am_lager, geliefert
          0.00 .F.  .  .
```

Geht ein Programm in dBase II davon aus, daß die Daten des letzten Satzes nach einem erfolglosen LOCATE verfügbar sind, so führt das zu einem bei der Konvertierung nicht erkennbaren Fehler.

Ein erfolgloses **FIND** liefert in dBase II für den Datensatzzeiger 0 und die Daten des logisch ersten Satzes.

```
. INDEX ON artikel TO artikel
00003 Sätze indiziert
. USE mp1 INDEX artikel
. FIND "Milch"
Wert nicht gefunden
. ? #, EOF
     0 .F.
. ? artikel, preis, am:lager, geliefert
Brot        1.05 .T. 08.03.88
```

In dBase III liefert der Datensatzzeiger die physisch letzte Satznummer plus 1, EOF() ist TRUE, und der Zugriff auf die Felder führt zur Ausgabe der Initialisierungswerte.

```
. USE mp1
. COPY STRUCTURE EXTENDED TO mp1s2
00005 Sätze kopiert
. USE mp1s2
. LIST
00001   ARTIKEL      C    10    0
00002   PREIS        N     8    2
00003   AM:LAGER     L     1    0
00004   GELIEFERT    C     8    0
00005   BEM          C    20    0
```

Bild 3 Strukturfile in dBase II

```
. USE
. USE mp1s2
. LIST
Satz #  FIELD_NAME FIELD_TYPE FIELD_LEN FIELD_DEC
     1  ARTIKEL    C                010       000
     2  PREIS      N                008       002
     3  AM:LAGER   L                001       000
     4  GELIEFERT  C                008       000
     5  BEM        C                020       000
. USE
. CREATE mp2 FROM mp1s2
Unzulässiger Feldname-- im Feld gefunden       3
* ':' im dritten Satz wird durch '_' ersetzt
. CREATE mp2 FROM mp1s2
. LIST STRUCTURE
Datenbankstruktur        - C:mp2.dbf
Anzahl der Datensätze    -       0
Letztes Änderungsdatum - 15.11.88
Feld    Feldname      Typ        Länge     Dez
   1    ARTIKEL       Zeichen      10
   2    PREIS         Numerisch     8       2
   3    AM_LAGER      Logisch       1
   4    GELIEFERT     Zeichen       8
   5    BEM           Zeichen      20
** Gesamt **                       48
```

Bild 4 Erstellen eines Datenbankfiles aus einem Strukturfile ▶

```
. INDEX ON artikel TO artikel
        3 Sätze indiziert
. USE mp1 INDEX artikel
. FIND Milch
Nicht gefunden
. ? RECNO(), EOF()
        4 .T.
. ? artikel, preis am_lager, geliefert
        0.00 .F.  .  .
```

Das Konvertierungsprogramm konvertiert deshalb Kommandofolgen der Art

```
FIND xyz
DO WHILE # = 0
```

in

```
*!! EOF() will be true if NO FIND, and RECNO()
*will equal BOTTOM, not 0
FIND xyz
DO WHIL (EOF() .OR. BOF())
```

Die Zeichen *!! leiten Kommentare ein, die beim Konvertieren hinzugefügt wurden.

Logische Konstanten werden in dBase II ohne Punkte, also nur als **N**, **T** usw. geschrieben. Dadurch können auch Speichervariablen, die nur Zeichen enthalten, als logischer Ausdruck verwendet werden. Betrachten wir dazu die Kommandofolge:

```
STORE N TO control
DO WHILE control
...
```

Sie wird konvertiert in

```
* !! Logical constant converted
STORE .N. TO control
DO WHILE control
...
```

In diesem Fall ist die Konvertierung korrekt. Betrachten wir ferner:

```
STORE N TO answer
...
INPUT "(Y/N)" TO answer
IF answer
...
```

INPUT in dBase III verlangt, logische Konstanten gemäß den Festlegungen, also in Punkte eingeschlossen, einzugeben. Das Prompt wird dadurch inhaltlich falsch, was aus der Sicht der Nutzeroberfläche als Fehler erscheint.

```
* !! Logical constant converted
STORE .N. TO answer
...
* !! There will be no automatic colon following
* this prompt string
INPUT "(Y/N)" TO answer
IF answer
...
```

Die meisten Probleme treten bei der Benutzung von **Speichervariablen** auf. Eine Speichervariable ist in dBase II global, das heißt, sie ist in allen Kommandofiles und auf der Dialogebene gültig. In dBase III ist eine Speichervariable lokal in dem File, in dem sie deklariert wird. Sie ist so lange gültig, bis das betreffende Kommandofile (Prozedurfile) abgeschlossen wird (CLOSE PROCEDURE, SET PROCEDURE oder RETURN aus einem Kommandofile). Bei der Konvertierung werden alle Variablen zu lokalen Variablen. Es wird eine erfolgreiche Konvertierung gemeldet. Bei der Abarbeitung der Kommandofiles kann es zu Fehlermeldungen durch den Interpreter von dBase kommen, etwa der Art

```
Variable not found
        ?
IF MADZAHL = 0
```

```
j
Mikroprozessortechnik
n
j
n
20,artikel
Artikel
10,preis
>Preis
n
12,geliefert
<Lieferdatum
```

Bild 5 dBase II-Format-File für Berichte

```
. USE mp1
. REPORT FORM mp1

          Seitennr.      1
          15.11.88
                                        Mikroprozessortechnik

          Artikel                  >Preis <Lieferdatum

          Butter                        2 12.02.89
          Brot                          1 08.03.88
          LΣ┳ ▌ΘΩ·                      0 01.03.89
```

Bild 6 Bericht bei Benutzung eines konvertierten REPORT-Format-Files

MADZAHL ist in diesem Beispiel eine Speichervariable, der in einem anderen Kommandofile ein Wert zugewiesen wurde. Für das aufgerufene Kommandofile ist die Variable *MADZAHL* undefiniert und die Fehlermeldung berechtigt. Die hier beschriebene Situation tritt bei der Konvertierung des *Adreßbeispiels* auf, das als Demonstrationsbeispiel zur Version 2.3B geliefert wurde.

Es können aber auch Fehler entstehen, die nicht zur Abarbeitung gemeldet werden. Dazu betrachten wir das folgende Beispiel: Gegeben sind zwei Kommandofiles *c1.prg* und *c2.prg* . In *c1* wird einer Variablen *option* ein Anfangswert zugewiesen. In *c2* wird dem Nutzer im Dialog die Möglichkeit gegeben, Optionen zu setzen. Das führt intern unter Umständen zur Veränderung der Variablen *option*. Nach der Rückkehr aus *c2* wird in *c1* mit den nunmehr aktuellen Werten weitergearbeitet. In dBase II arbeitet dieses Programm wie beschrieben. In dBase III wird bei der Abarbeitung kein Fehler gemeldet, denn der Variablen *option* wird in *c2* möglicherweise ein Wert zugewiesen. Es wird nicht, wie im ersten Beispiel, auf einen Wert zugegriffen. Da *option* in *c1* und *option* in *c2* verschiedene Variable sind, gelangt der in *c2* eingelesene Wert nie nach *c1*, so daß in *c1* immer mit der Voreinstellung aus *c1* gearbeitet wird. Derartige Fehler sind, besonders in fremden Programmen, schwer zu finden.

In den meisten Fällen hilft die Vereinbarung der global benötigten Variablen als *PUBLIC* in allen betroffenen Files.

REPORT-FILES

Berichtsformate unterscheiden sich in der Positionierung der Spaltenüberschriften. dBase II gestattet eine Zentrierung, eine links- oder rechtsbündige Anordnung. Dazu werden die Überschriften mit dem Zeichen < für linksbündig und > für rechtsbündig eingeleitet. Diese Zeichen werden nicht konvertiert. Sie erscheinen in den Überschriften der dBase III-Berichte.

Logische Konstanten werden auch in Report-Files von dBase II abgelegt. Auch hier wird ein *J* aus deutschsprachigen dBase II-Versionen nicht korrekt konvertiert, wodurch die gesamte Berichtsform durcheinander gerät. Bild 5 zeigt als Beispiel ein Report-File, in dem die Frage nach der Seitenüberschrift mit *j* beantwortet wurde. Das Konvertierungsprogramm meldet den Fehler **Warning – REPORT FORM EMPTY**, was offenbar falsch ist. Ändert man alle logischen Konstanten *j* im Report-File vor der Konvertierung in *y*, so wird das File fehlerfrei konvertiert. Bild 6 zeigt den unter dBase III erzeugbaren Bericht.

Memory-Files und Bildschirmmasken

Beide Filetypen (*.mem, .fmt*) konnten in allen getesteten Fällen korrekt konvertiert werden. Für Bildschirmmasken, die nur aus Kommentaren und @-Kommandos bestehen und keine Formatklauseln (USING, PICTURE) enthalten, ist keine Konvertierung erforderlich.

Indexfiles

Indexfiles werden nicht direkt konvertiert. Aus dem dBase II-Indexfile wird der Indexausdruck entnommen und in das Kommando INDEX ON <ausdr> TO <indexfile> eingesetzt. Es bildet den Inhalt eines Kommandofiles mit der Namenserweiterung *.rx.*, welches für jedes zu konvertierende Indexfile erstellt wird. Unter dBase III sind diese Kommandofiles anzuarbeiten.

Literatur

/1/ Grafik, W.: dBASE III. Berlin: Verlag Technik; in Vorbereitung

✉ KONTAKT ☎

Humboldt-Universität Berlin, Sektion Mathematik, Bereich IV, PSF 1297, Berlin, 1086; Tel. 2093 23 48

Zeichnung: Steger



Na, Meister, wieviel Anschläge pro Minute schaffen wir denn schon beim Programmieren?

Automatische Generierung von BASIC-INLINE-POKEs

Christian Hanisch, Berlin

Unsere Beitragsfolge „Werkzeuge zur hardwarenahen Programmierung in höheren Programmiersprachen auf 8-Bit-Computern“, die wir in MP 10/1988 begannen und in 11/1988 fortsetzten, behandelt in ihrem dritten Teil die automatische Generierung von BASIC-INLINE-POKEs.
INLINE-Assembler ist in der Praxis der Anwendung höherer Programmiersprachen bei Personalcomputern zum Hilfsmittel betriebssystem- und/oder hardwarenaher Problemlösungen (hrp=hardware related programming /1/) geworden. Für Turbo-Pascal z. B. steht mit dem Programm PMLINK /2/ eine gute Variante der INLINE-Routinen-Generierung zur Verfügung.
Auch Basic-80-Programme können durch Assemblerunterprogramme sowohl bezüglich Effizienz als auch in ihrem Anwendungsspektrum wesentlich erweitert werden. Die Methode der „gepokten“ Assemblerbefehle ist in vielen Basic-Dialekten unterstützt und sowohl im Interpreter- als auch im Compilersystem anwendbar.

Ausgangsbasis der hier dargelegten Methode ist eine COM-Datei. Das hat den Vorteil, daß neben reinen Assemblerroutinen z. B. ebenfalls *kleine* Fortran-Unterprogramme unter anderem über einen Aufrufvermittler, mit dem eventuell einige Parameteranpassungen realisiert werden, entsprechend Bild 1 zusammengelinkt und in BASIC-INLINE-POKEs umgewandelt bzw. mit dem hier vorgestellten RUN-Time-Lader (Bild 6) geladen werden können.
Die Fortran-Subroutine FROUT.REL kann damit in FROUT.OVX eingebunden als Unterprogramm ab der Adresse 0A500H benutzt werden.

Anmerkung: Im Basic-Compilersystem, z. B. mit dem Rahmenprogramm FROUTASS, wird dagegen FROUT so eingebunden:

```
B>:BASC FROUTASS, TTY:=FROUTASS/N/E/X<CR>
B>A:BASLINK FROUTASS/N, FROUTASS, A:BASLIB/S,
[/P:A500,] [ASSVER,]FROUT, A:FORLIB/S/E<CR>
```

Die Transformation von internem Binärcode aus einer COM-Datei in eine „INCLUDE“-Datei für Basic-80, die man mit einem Textverarbeitungsprogramm (z. B. dem Editor von Turbo-Pascal) als GOSUB-Routine einfügen kann, wird in dem nachfolgend mitgeteilten Turbo-Pascal-Programm YPOKEBAS realisiert.
Als Anwendungsbeispiel soll in /3/ mittels YPOKE als dBase II-INLINE-Routine gezeigte Assemblerunterprogramm SPACE.CMD dienen. Mit diesem Unterprogramm können der noch verfügbare Speicherplatz bzw. die noch freien DIRectory-Entries auf der Diskette des im Parameter mitgeteilten Laufwerkes für eine Aufzeichnungsblockgröße von 1 oder 2 KByte ermittelt werden. Die Kenntnis dieser Angaben zur RUN-Time ist für viele Anwendungen im Zusammenhang mit Dateiarbeit nützlich. Das Basic-Demonstrationsprogramm SPACEDMO allein schon ist zur wahlweisen Anzeige des noch freien Platzes auf Diskette und im DIRectory eines beliebigen Laufwerkes analog dem STAT-Kommando unter CP/M mit einer weiteren Information einsetzbar.
Das Beispiel ist mit dem Startprogramm EXEC aus /3/ soweit abgestimmt, daß SPACEDMO auch unter EXEC laufen könnte.
Der Basic-Quelltext im Bild 3 gibt zusätzlich einige weniger bekannte interne Informationen bezüglich der Probleme bei Parameterübergabe an Unterprogramme.
Die Ladeadresse sei im Assemblerunterprogramm auf 0A40FH (41999) festgelegt. Die Form der Parameterübergabe ist für jeden Anwendungsfall geeignet festzulegen. Hier werden aus Kompatibilitätsgründen die dBase II-Konventionen benutzt.
Das Testen vor der Umwandlung in BASIC-INLINE-POKEs würde im Interpretermodus wie in Bild 4 dargestellt ablaufen.
Hier würde sich dann anschließen:

```
B>A:YPOKEBAS SPACE.OVX SPACE.BAS<CR>
```

Die Datei SPACE.BAS enthält unter den obengenannten Voraussetzungen die Basic-Data-Anweisungen und INLINE-POKEs für das Unterprogramm (GOSUB-Routine SPACE.BAS). Die Anwendung des SPACE-Unterprogramms zeigt das im Bild 3 mitgeteilte Demonstrationsprogramm SPACEDMO.BAS.

Den Quelltext des Turbo-Pascal-Programms YPOKEBAS.PAS enthält das Bild 5.
Falls die mit YPOKEBAS erstellte Folge von DATA-Anweisungen und POKE-Befehlen einen größeren Umfang annimmt, kann die hier vorgestellte Methode aus Speicherplatzgründen vom Interpreter nicht mehr bewältigt werden. Dieser Fall trifft unter anderem bei der zu Beginn erwähnten Fortran-Applikation FROUT.OVX (19 * 128 Byte) zu.
In diesem Falle bietet sich eine Methodik der dynamisch nachladbaren externen Unterprogramme an. Eine für YPOKEBAS als Grundlage dienende COM-Datei, hier z. B. SPACE.OVX oder FROUT.OVX, kann dynamisch zur RUN-Time aus dem Basic-Programm heraus auf die Ladeadresse (z. B. 0A40FH bzw. 0A500H) nachgeladen werden.
Dazu benötigt man einen **RUN-Time-Lader,** der selbst auch als Assemblerunterprogramm ausgelegt ist, welcher mit YPOKEBAS als DATA-Befehlsfolge – wie bereits beschrieben – integriert werden kann. Weitere Unterprogramme können als OVX-Dateien von Diskette je nach Bedarf – auch in dynamischer Überlagerungsstruktur – nachgeladen werden.
Für ein nach den Methoden der strukturierten Programmierung konzipiertes Programmpaket haben dynamisch nachladbare Unterprogramme – es ist hier an betriebssystem- und/oder hardwarenahe Assembler-Komponenten gedacht – bezüglich Programmwartung und Portabilität wesentliche Vorteile.
Ein nicht seriell wiederverwendbarer **RUN-Time-Lader** XPOLAD.MAC ist in Bild 6 gezeigt.
Dem interessierten Leser wird empfohlen, diesen Assemblerquelltext nach den obigen Ausführungen mit YPOKEBAS in eine GOSUB-Routine XPOLAD.BAS zu überführen. Dazu müssen die Zeilen 1000 bis 1014 des Bildes 2 durch die GOSUB-Routine XPOLAD.BAS und die Zeile 100 des Bildes 3 durch:

```
100 GOSUB 1000
101 XPOLAD=92
102 DSN$="SPACE OVX":LAD%=&HA40F:
  <<^PJ>> EAD%=LAD%+(5+1)*128
103 IF EAD%>=PEEK(7)*256+PEEK(6) <<^PJ>>
  THEN PRINT "CANCEL because Kill BDOS":END
104 CALL XPOLAD (DSN$, LAD%, EAD%)
105 IF LAD%=0 OR EAD%=0 THEN <<^PJ>>
  PRINT "Error → CANCEL":  END
```

ersetzt werden.
Im Basic-Compilersystem wird XPOLAD im CALL-Statement als externe Referenz vom

Bild 1 Arbeitsschritte zur Erstellung eines für BASIC-INLINE-POKEs bzw. zur RUN-Time-Nachladung vorbereiteten COM-Files am Beispiel von Fortran-80

```
B>A:F80 FROUT, TTY:=TTY:/N<CR>
      subroutine FROUT(X)
      integer X
      real Y
      Y=X
      X=sqrt(Y)
      return
      end
<<^Z>>
B>A:M80 ASSVER, TTY:=TTY:/Z/X<CR>
         .phase ØA5ØØh
         extrn FROUT
ASSVER:: call  FROUT
         ret
         end
<<^Z>>
B>A:L80 /P:A5ØØ, ASSVER.REL, <<^E>>
FROUT.REL, A:FORLIB.REL/S, /E<CR>
B>A:POWER SAVE FROUT.OVX A5ØØ 19<CR>
```

Bild 2 Assemblerunterprogramm SPACE.BAS als BASIC-POKEs generiert mit YPOKEBAS

```
1ØØØ REM SPACE.BAS vom: Ø2.Ø3.88
1ØØ1 REM Generiert mit YPOKEBAS aus der Datei: SPACE.OVX
1ØØ2 REM Aufruf im Anwenderprogramm ( %INCLUDE SPACE.BAS ):
1ØØ3 REM <Zeile> [GOSUB 1ØØØ:] <name>=&HA4ØF:
 CALL <name> [(<Arg.-Liste>)]
1ØØ4 DATA 229,17,188,165,1,5,Ø,237,176,58,4,Ø,5Ø,2Ø4,165,23Ø,
15,14,13,2Ø5,5,Ø,58,188,165,225,254,5,21Ø,173,165,254,4,218,
173,165,229,42,19Ø,165,17,32,32,237,82,225,2Ø2,231,164,35,34,
2Ø2,165,126,2Ø3,175,214,65,23Ø,15,95,14,14,2Ø5,5,Ø,14,31,2Ø5,
5
1ØØ5 DATA Ø,17,7,Ø,25,78,35,7Ø,3,237,67,195,165,33,Ø,Ø,34,197,
165,34,193,165,17,241,165,14,26,2Ø5,5,Ø,62,63,5Ø,2Ø9,165,17,
   ...                    ...                 ...
1Ø11 DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
1Ø12 RESTORE 1ØØ4
1Ø13 FOR I1ØØ4%=Ø TO 639:READ P1ØØ4%:
 POKE &HA4ØF+I1ØØ4%,P1ØØ4%:NEXT I1ØØ4%
1Ø14 RETURN
```

```
5 REM "SPACEDMO.BAS"                                    Datum: Ø3.Ø2.88
6 REM Demostrationsprogramm fuer den Einsatz
7 REM der mit    YPOKEBAS.COM    erstellten
8 REM POKEd-Assemblerroutine zum Auslesen des noch
9 REM freien Speicherplatzes / freie DIRectory-Entries
1Ø REM auf der Diskette des Arbeitslaufwerkes <d>.
11 REM Parameter-/Ergebnis-Uebergabe:
12 REM     store "B"+"DIR" to  Mparm
13 REM z.B.  ¦Ø4¦42¦44¦49¦52¦    ---> ¦Ø4¦2Ø¦31¦33¦36¦
14 REM A>MBASIC B:SPACEDMO/M:&HA4ØE<CR>
1ØØ GOSUB 1ØØØ  ' Assemblerroutine --> 41999 poken
11Ø PRINT CHR$(12)+CHR$(27)+CHR$(14Ø)+CHR$(128);
12Ø PRINT "ENTER Arbeitslaufwerk ('<d>')  A..P  ===>";
13Ø A$=INPUT$(1):I%=ASC(A$):IF I%>95 THEN A$=CHR$(I% AND 95)
14Ø PRINT CHR$(27)+CHR$(129)+CHR$(14Ø)+"Arbeitslaufwerk ";
15Ø PRINT A$: PRINT CHR$(27)+CHR$(131)+CHR$(135)+"===> ";
16Ø PRINT "Freier Platz auf Diskette im Arbeitslaufwerk:"
3ØØ ASSUP5=-23537          ' ( Entry: &HA4ØF entspricht 41999 )
4ØØ REM Die Zuweisung ASSUP5=&HA4ØF wird in CONECT erledigt
41Ø REM Echte Adresse von A$ nach AAM% aufbereiten:
42Ø   AAM%=Ø
43Ø   AA%=VARPTR(AAM%)
44Ø   poke AA%, peek(VARPTR(A$)+1)
45Ø   poke AA%+1,peek(VARPTR(A$)+2)
46Ø AXD%=&HFØ     ' Die Parameter werden ab 24Ø abgelegt
47Ø poke AXD%,&HØ4          ' Laenge  nach 24Ø  ablegen
48Ø poke AXD%+1,peek(AAM%) ' Lw.  nach   241  ablegen
49Ø poke AXD%+2,&H2Ø: poke AXD%+3,&H2Ø
5ØØ poke AXD%+4,&H2Ø ' Drei Blanks --> 242-244 ablegen
51Ø GOSUB 8ØØ
52Ø AYD%=&HF2  ' Zum Abholen des Ergebnisses ab 242-244
53Ø MPARM$=CHR$(peek(AYD%))+CHR$(peek(AYD%+1))
535 MPARM$=MPARM$+CHR$(peek(AYD%+2))
54Ø PRINT:PRINT "F r e i e  K a p a z i t a e t  auf Lw. ";
545 PRINT CHR$(134)+A$;
55Ø PRINT CHR$(132)+": "+MPARM$+CHR$(134)+" KByte"+CHR$(132)
56Ø REM MPARM erneut aufbereiten fuer naechsten Aufruf:
57Ø poke AXD%,&HØ4
58Ø poke AXD%+1,peek(AAM%)
59Ø poke AXD%+2,&H44: poke AXD%+3,&H49: poke AXD%+4,&H52
6ØØ DEF USR1=&HEØ
61Ø IF peek(&HA4ØF)=229 THEN  X%=USR1(Ø) ELSE GOSUB 8ØØ
62Ø REM Falls ASSUP5 auf ØA4ØFh steht, enthaelt X% spaeter
63Ø REM die Zahl 239 bzw. beim Interpreter wieder Ø.
64Ø REM Der Intgerpreter weist stets das Argument zu, d.h.
65Ø REM X%=USR1(55) ist identisch mit X%=55.
66Ø MPARM$=CHR$(peek(AYD%))+CHR$(peek(AYD%+1)).
665 MPARM$=MPARM$+CHR$(peek(AYD%+2))
67Ø PRINT CHR$(13)CHR$(1Ø);
68Ø PRINT "Anzahl freier DIRECTORY-Entries auf Lw.  ";
685 PRINT A$+" : "+MPARM$+CHR$(13)CHR$(1Ø)
7ØØ PRINT: PRINT "Programm beenden    (j/N) ? ";
71Ø B$=INPUT$(1)
72Ø IF B$="J" OR B$="j" THEN END ELSE GOTO 11Ø
73Ø REM  ===========  Programm-Ende ==========
8ØØ CONECT=&HEØ    ' Routine auf 224 poken
81Ø poke &HEØ,&H21: poke &HE1,&HFØ: poke &HE2,&HØØ
82Ø poke &HE3,&HE5
83Ø poke &HE4,&HCD: poke &HE5,&HØF:,poke &HE6,&HA4
84Ø poke &HE7,&HE1: poke &HE8,&H2B: poke &HE9,&HC9
85Ø   REM  ld hl,ØFØh ; 24Ø --> hl
86Ø   REM  push hl
87Ø   REM  call ØA4ØFh
88Ø   REM  pop hl
89Ø   REM  dec hl
9ØØ   REM  ret
91Ø CALL CONECT
92Ø RETURN
93Ø REM Hier die Datei SPACE.BAS einbinden, die aus der
94Ø REM Datei  SPACE OVX  mit  YPOKEBAS  erstellt wurde
95Ø REM (Einkopieren z.B. mit WordStar <<<^KR>> ).
96Ø REM Die benoetigte Assemblerroutine kann auch mit:
97Ø REM B>A:EXEC SPACEDMO.BAS/M:&HA3FF<CR>
98Ø REM und "MBASIC" anstelle "DBASE" geladen·werden.
1ØØØ REM SPACE.BAS vom: Ø3.Ø2.88
1ØØ1 REM Generiert mit YPOKEBAS aus der Datei: SPACE.OVX
       ...     SPACE.BAS aus Bild 1       ...
1Ø13 FOR I1ØØ4%=Ø TO 639:READ P1ØØ4%:
 POKE &HA4ØF+I1ØØ4%,P1ØØ4%:NEXT I1ØØ4%
1Ø14 RETURN
```

Bild 3 Demonstrationsprogramm für INLINE-POKEs "SPACE.BAS"

```
B>A:M8Ø<CR>
* SPACE,SPACE=SPACE.MAC/Z/X/M/C<CR>
B>A:L8Ø /P:A4ØF,SPACE/E<CR>
B>A:MBASIC  /M:41998<CR>
load "B:SPACEDMO"
run
 ...
system
B>A:POWER SAVE SPACE.OVX A4ØF 5<CR>
```

Bild 4 Bild 4 Test vor der Umwandlung in BASIC-INLINE-POKEs

```
Ø PROGRAM YPOKEBAS;
Ø (***************************************************)
Ø (* Fuer eine [Assembler-]COM-Datei wird eine POKE-Be-    *)
Ø (* fehlsroutine fuer BASIC-8Ø erzeugt, die das Programm  *)
Ø (* beim Aufruf mit GOSUB <zeile> als INLINE-Code auf die *)
Ø (* LINK-Adresse, z.B. /P:A4ØF (41999), uebertraegt.      *)
Ø (* Das Programm erfordert die TURBO-Version 3.ØØx.       *)
Ø (***************************************************)
Ø (*$V- *)
Ø CONST K1=' DATA ';
Ø TYPE Hilftext  = STRING[8Ø];
Ø      setofchar = SET OF CHAR;
Ø VAR  f: FILE;  {INPUT : COM-Datei als Untyped File}
Ø      p: TEXT;  {OUTPUT: dBASE-CMD-Datei }
Ø      i,k,m,znr,nrblocks,oldlabel,
Ø      newlabel,trmstr,nrbyte,intadr    : INTEGER;
Ø      hpart,lpart                      : BYTE;
Ø      Adr                              : REAL;
Ø      znra,znra1,parkstr               : STRING[6];
Ø      Name1,Name2                      : STRING[14];
Ø      Befehl                           : STRING[3];
Ø      Datum                            : STRING[8];
Ø      ZEILE                            : STRING[255];
Ø      Puffer                           : ARRAY[Ø..127] OF BYTE;
Ø      phase                            : hilftext;
Ø
Ø (*--------------------------------------------*)
Ø FUNCTION HEX(I:BYTE):CHAR;
Ø BEGIN  hex:=CHAR(48+i+7*ord(i>9)) END;
Ø (*--------------------------------------------*)
Ø (*$I LIESWORD.INC                                     *)
Ø (* Quelle Function lieszchn, Procedure lieswort:     *)
Ø (* Albuschat, F.: Eingabe unter Turbo-Pascal. Muenchen: *)
Ø (* Francis-Verlag, mc (1987)2  S. 62                 *)
Ø (* ------------------------------------------- *)
Ø FUNCTION HEXIN: REAL;
Ø CONST hexset: ARRAY[Ø..15] OF CHAR='Ø123456789ABCDEF';
Ø VAR {phase: hilftext; <-- im Hauptprogramm }
Ø     XP  : ARRAY[1..4] OF BYTE;  I,J : INTEGER;
Ø BEGIN
1   Writeln;
1   REPEAT
2     Write(' 4-stellige hexadezimale Ladeadresse'+
2       ' (<Adr_HEX>): ',phase:4,#8#8#8#8);
2     lieswort(['Ø'..'9','A'..'F','a'..'f'],4,phase);
2   UNTIL length(phase)=4;
1   FOR I:=1 TO 4 DO BEGIN
2     phase[i]:=upcase(phase[i]);j:=Ø;
2     WHILE hexset[j]<>phase[I] DO j:=succ(j);
2     xp[i]:=j;
2   END;
1   HEXIN:=4Ø96.Ø*XP[1]+256.Ø*XP[2]+16.Ø*XP[3]+XP[4]
1 END;
Ø
Ø (*------------------  Hauptprogramm  ------------------*)
Ø BEGIN
1   clrscr; phase:='A4ØF'; znr:=1ØØØØ;
1   Writeln(#1Ø#1Ø#134+
1   '[Assembler-]COM-File ---> BASIC-INLINE-POKEs'#132);
1   Writeln(
1   ' ==== 8ØØØ ========= V.m=3.Ø ================');Writeln;
1   Writeln(' B>A:M8Ø <filename_REL>,<filename_PRN>='+
1   '<filename_MAC>/M<CR>');
1   Writeln(' B>A:L8Ø /P:<Adr_HEX>,<filename_REL>/E<CR>');
1   Writeln(' B>A:POWER SAVE <filename_COM>.<type>'+
1   '  <Adr_HEX>  <anz><CR>');gotoxy(1,1Ø);
1   Writeln(#134'  B>A:YPOKEBAS   <filename_COM> '+
1   ' <filename_BAS><CR>'#132);
1   IF ParamCount<2 THEN BEGIN
2     Writeln('*** Ein- und Ausgabedatei in der Aufruf'+
2     'zeile angeben'#13#1Ø' ---> Noch einmal !'); HALT END
1                        ELSE BEGIN
2     Name1:=ParamStr(1); Name2:=ParamStr(2);
2     IF POS('.',Name1)=Ø THEN Name1:= Name1+'.OVX';
2     IF POS('.',Name2)=Ø THEN Name2:= Name2+'.BAS';
2     {$I-} ASSIGN(f,Name1); RESET(f); {$I+}
2     IF IORESULT<>Ø THEN BEGIN
3       Writeln ('*** Eingabedatei ',Name1,
3        ' nicht gefunden --> ABBRUCH !');HALT
3     END
2   END;
1   nrblocks:=filesize(f);
1   IF nrblocks=Ø THEN BEGIN
2     writeln('*** Eingabedatei ist leer ---> CANCEL'); halt
2   END;
1   ASSIGN(p,Name2); REWRITE(p);
1   GotoXY(16,1Ø);clreol;
1   writeln(Name1:16,name2:16,#132);writeln;
1   write(' aktuelles Datum (dd.mm.jj): ');
1   Readln(Datum);
1   adr:=hexin;  writeln;
1   write(' Zeilennummer der BASIC-POKE-Routine (<=3ØØØØ)  : ');
1   readln(znr);writeln;
1   str(znr,znra1);
1   zeile:=znra1+' REM '+name2+' vom: '+Datum;
1   writeln(p,zeile);writeln(zeile);
1   znr:=succ(znr);str(znr,znra);
1   zeile:=znra+' REM Generiert mit YPOKEBAS aus der Datei: '+
1   name1; writeln(p,zeile); writeln(zeile);
1   znr:=succ(znr);str(znr,znra);
1   zeile:=znra+' REM Aufruf im Anwenderprogramm'+
1   ' ( %INCLUDE '+Name2+' ):';
1   writeln(p,zeile);writeln(zeile);
1   znr:=succ(znr);str(znr,znra);
1   zeile:=znra+' REM <Zeile> [GOSUB '+znra1+':] <name>='+
1     '&H'+phase+':'#$ØA#$ØD' CALL <name> [(<Arg.-Liste>)]';
1   str(succ(znr),znra1);
1   oldlabel:=Ø;
1   nrbyte:=pred(nrblocks*128);
1   str(nrbyte,parkstr);
1   newlabel:=249;trmstr:=length(zeile);
1   FOR i:=1 TO nrblocks DO BEGIN
2     blockread(f,Puffer,1);
2     m:=Ø;
2     REPEAT
3       k:=Puffer[m];
```

```
3           str(k,Befehl);
3           IF newlabel>248 THEN BEGIN
4                 writeln(p,zeile);
4                 writeln(copy(zeile,succ(oldlabel),trmstr));
4                 znr:=succ(znr);str(znr,znra);oldlabel:=Ø;
4                 zeile:=znra+kl+befehl END
3           ELSE IF trmstr>59 THEN BEGIN
4                  Zeile:=Zeile+',';
4                  write(copy(zeile,succ(oldlabel),SUCC(trmstr)));
4                  oldlabel:=length(zeile);
4                  zeile:=zeile+#$ØA#$ØD+befehl; END
3                             ELSE zeile:=zeile+','+befehl;
3           newlabel:=length(zeile); trmstr:=newlabel-oldlabel;
3           m:=succ(m);
3         UNTIL m=128;
2       END;
1       writeln(p,Zeile);
1       writeln(copy(Zeile,succ(oldlabel),trmstr));
1       znr:=succ(znr);str(znr,znra);
1       zeile:=znra+' RESTORE '+znral;
1       writeln(p,zeile);writeln(zeile);
1       znr:=succ(znr);str(znr,znra);
1       Zeile:=znra+' FOR I'+znral+'%=Ø TO '+parkstr+
1        ':READ P'+znral+'%:'#$ØA#$ØD' POKE &H'+phase+
1        '+I'+znral+'%,P'+znral+'%:NEXT I'+znral+'%';
1       Writeln(p,Zeile); Writeln(Zeile);
1       znr:=succ(znr);str(znr,znra);
1       zeile:=znra+' RETURN';
1       Writeln(p,zeile); Writeln(zeile);
1       CLOSE(p);
1       intadr:=round(ORD(adr>$7FFF)*-65535.Ø+adr);
1       Write(#13#1Ø'Das POKEd Unterprogramm steht ab: '+phase+
1        (',intadr,')'#13#1Ø'                                bis: ');
1       adr:=adr+nrbyte;
1       intadr:=round(ORD(adr>$7FFF)*-65535.Ø+adr);
1       hpart:=hi(intadr);lpart:=lo(intadr);
1       write(hex(hpart SHR 4)+hex(hpart AND $ØF)+
1             hex(lpart SHR 4)+hex(lpart AND $ØF));
1       writeln('  (',intadr,')');
1       REPEAT UNTIL keypressed
1     END.
```

***Bild 5** Generierungsprogramm YPOKEBAS.PAS (erzeugt BASIC-INLINE-POKEs aus einem COM-File)*

***Bild 6** Basic-80-RUN-Time-Lader für externe Unterprogramme* ▶

```
        .z8Ø
        title XPOLAD.MAC  ( RUN-Time-Lader fuer BASIC-8Ø )
; B>A:M8Ø xpolad,lst:=xpolad/z/x/c/m<CR>
; B>A:L8Ø /P:CØØØ,xpolad/E<CR>  ( auf beliebige Adr. )
; B>A:POWER SAVE xpolad.ovx CØØØ 2<CR>
; B>A:YPOKEBAS xpolad.ovx xpolad.bas<CR>     ===> XPOLAD.BAS
; Aufruf:
; 1Ø GOSUB <Zeile von XPOLAD.BAS>  ( z.B. <Zeile>::= 1ØØØ )
; 11 XPOLAD=92   <Startadresse XPOLAD in BASE-Page ab 5Ch >
; 12 DSN$="SPACE   OVX"                ( genau 11 Zeichen )
; 13 LAD%=&HA4ØF: EAD%=LAD%+(5+1)*128
; 14 IF EAD%>=PEEK(7)*256+PEEK(6) THEN PRINT "Kill BDOS":END
; 15 CALL XPOLAD (DSN$,LAD%,EAD%)
; 16 IF LAD%=Ø THEN < Datei not found  ¦  LEN(DSN$)<>11 >
; 17 IF EAD%=Ø THEN < Datei zu gross >
; 18 REM Achtung: XPOLAD ist nicht seriell wiederverwendbar!
; 19 REM SPACE.OVX sei auf ØA4ØFh (41999) zu laden.
; Hinweis: Die Assemblerroutine SPACE.OVX wird folgender-
; =======  massen uebersetzt, gelinkt und abgelegt.
; B>A:M8Ø SPACE,LST:=SPACE/Z/X/C/M<CR>
; B>A:L8Ø /P:A4ØF,SPACE/E<CR>
; B>A:POWER SAVE SPACE.OVX A4ØF 5<CR>
          .phase 5Ch
XPOLAD::ld    (parm1),hl   ; Adr. DSN$
        ld    (parm2),de   ; Adr. LAD%
        ld    (parm3),bc   ; Adr. EAD%
        ld    a,(hl) ; Reg. hl zeigt auf <laenge> von DSN$
        cp    11      ; ¦<laenge> ¦ 2 Byte Adr. Text DSN$ ¦
        jp    nz,XXØ ; Reg. de enthaelt Adr. von LAD%
        ld    c,25        ; Aktuelles Lw. ermitteln
        call 5
        inc   a
        ld    (92),a
        ld    iy,(parm1)
        ld    l,(iy+1)    ; Adr. d. Textes von DSN$ --> hl
        ld    h,(iy+2)
        ld    de,92+1     ; FCB-OVERLAY XPOLAD: 5Ch bis 7Fh
        ld    bc,11
        ldir
        ld    a,Ø
        ld    (92+12),a
        ld    (92+32),a
        ld    de,92       ; OPEN fuer Datei aus DSN$
        ld    c,15        ; auf aktuellem Lw.
        call 5
        cp    ØFFh        ; Datei nicht gefunden --> CANCEL
        jr    z,XXØ
        ld    ix,(parm2)
        ld    e,(ix)
        ld    d,(ix+1)
        push de
XX2:    pop   de          ; DMA-Adr.         := Adr. aus LAD%
        push de
        ld    c,26        ; bzw. DMA-Adr. := DMA-Adr. + 128
        call 5
        ld    de,92       ; Sequentielles Lesen von Sektoren
        ld    c,2Ø        ; zu je 128 Byte nach TPA (DMA-Adr.)
        call 5
        pop   hl
        and   a
        ret   nz          ; EOF erreicht
        ld    bc,128      ; Vorbereitung Lesen naechsten Sektor
        add   hl,bc
        push hl
        pop   de
        ld    iy,(parm3)
        ld    l,(iy)      ; Datei zu gross ?
        ld    h,(iy+1)
        or    a
        sbc   hl,bc
        or    a
        sbc   hl,de
        jr    nc,XX3
        ld    iy,(parm3)
        ld    (iy),Ø      ; Datei ist zu gross
        ld    (iy+1),Ø
        ret
XX3:    push de
        jr    XX2
XXØ:    ld    ix,(parm2)
        ld    (ix),Ø
        ld    (ix+1),Ø
        ret     ; LEN(DSN$)<>11  ¦  DSN$ not found
parm1:  defw 5858h ; Adr. DSN$
parm2:  defw 5858h ; Adr. LAD%
parm3:  defw 5858h ; Adr. EAD%
        end
```

```
1ØØØ REM XPOLAD.BAS vom: Ø2.Ø3.88
1ØØ1 REM Generiert mit YPOKEBAS aus der Datei: XPOLAD.OVX
1ØØ2 REM Aufruf im Anwenderprogramm ( %INCLUDE XPOLAD.BAS ):
1ØØ3 REM <Zeile> [GOSUB 1ØØØ:] <name>=&HØØ5C:
 CALL <name> [(<Arg.-Liste>)]
1ØØ4 DATA 34,238,Ø,237,83,24Ø,Ø,237,67,242,Ø,126,254,11,194,225,
Ø,14,25,2Ø5,5,Ø,6Ø,5Ø,92,Ø,253,42,238,Ø,253,11Ø,1,253,1Ø2,2,
17,93,Ø,1,11,Ø,237,176,62,Ø,5Ø,1Ø4,Ø,5Ø,124,Ø,17,92,Ø,14,15,
2Ø5,5,Ø,254,255,4Ø,69,221,42,24Ø,Ø,221,94,Ø,221,86,1,213,2Ø9
1ØØ5 DATA 213,14,26,2Ø5,5,Ø,17,92,Ø,14,2Ø,2Ø5,5,Ø,225,167,192,
1,128,Ø,9,229,2Ø9,253,42,242,Ø,253,11Ø,Ø,253,1Ø2,1,183,237,
66,183,237,82,48,13,253,42,242,Ø,253,54,Ø,Ø,253,54,1,Ø,2Ø1,
213,24,198,221,42,24Ø,Ø,221,54,Ø,Ø,221,54,1,Ø,2Ø1,88,88,88,88
1ØØ6 DATA 88,88
1ØØ7 RESTORE 1ØØ4
1ØØ8 FOR I1ØØ4%=Ø TO 151:READ P1ØØ4%:
 POKE &HØØ5C+I1ØØ4%,P1ØØ4%:NEXT I1ØØ4%
1ØØ9 RETURN
```

***Bild 7** Basic-80-RUN-TIME-Lader aus Bild 6 als GOSUB-Routine (generiert mit YPOKEBAS und auf 152 Byte gekürzt)*

Linker aufgelöst, so daß der Assembler-Quelltext aus Bild 6 nach XPOLAD.REL übersetzt, dazugelinkt werden muß. Das trifft ebenfalls für CONECT aus Bild 3 zu. Man hat im Ergebnis mit den Programmen XPOLAD-.BAS und YPOKEBAS.COM einen praktisch sofort nutzbaren Satz von Werkzeugen zur dynamischen [Assembler-]Unterprogrammtechnik in Basic-80 zur Verfügung.

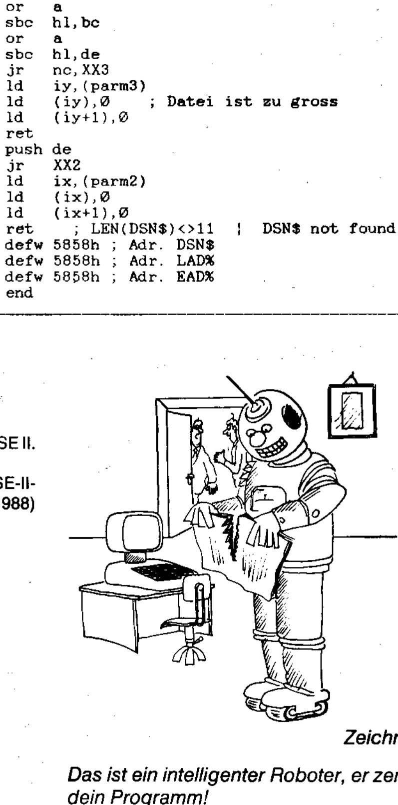
Literatur

/1/ Grafik, W.: Hardwarenahe Programmierung in höheren Programmiersprachen. edv-Aspekte, Berlin 6 (1987) 4, S. 54

/2/ Nerz, H.: Assemblerroutinen in Turbo-Pascal. mc, München (1986) 6, S. 78

/3/ Hanisch, Ch.: EXEC – ein Startprogramm für dBASE II. Mikroprozessortechnik, Berlin 2 (1988) 11, S. 331

/4/ Hanisch, Ch.: Umwandeln von COM-Files in dBASE-II-INLINE-POKEs. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 307



Zeichnung: Steger

Das ist ein intelligenter Roboter, er zerreißt gerade dein Programm!



JOHANN CARL FRIEDRICH GAUSS

*** 1777 Braunschweig, † 1855 Göttingen**

Nach dem Tode des fast 78jährigen Gauß wurde im Auftrag des damals regierenden Königs Georg V. von Hannover eine Gedenkmedaille mit dem Bildnis von Gauß geprägt, die ihn als „Fürst unter den Mathematikern" bezeichnet. Jahre zuvor war ihm auf Anregung Humboldts die höchste Auszeichnung des preußischen Staates, der *Pour le mérite*, verliehen worden. Diese hohen Ehrungen bestätigen den Umstand, daß Gauß bereits zu Lebzeiten unbestritten auf der ganzen Welt als höchste Autorität im Bereich der Mathematik und ihrer Anwendungen galt. Schon seine schätzungsweise 7000 Briefe – geschrieben in einem halben Dutzend verschiedener Sprachen – lassen etwas von seinem internationalen Rang ahnen!

Tatsächlich waren seine mathematischen Neuentdeckungen mitunter so kühn, daß er diese der Öffentlichkeit gar nicht mitzuteilen wagte, weil er das „Geschrei der Böotier" fürchtete (wie es 1829 in einem Brief an F. W. Bessel heißt); so mußte er es beispielsweise erleben, daß ihm N. I. Lobatschewskij und W. Bólyai mit der Veröffentlichung der *Nichteuklidischen Geometrie* zuvorkamen, obwohl er diese bereits 20 Jahre vorher durchdacht und die entsprechenden Aufzeichnungen im Schubfach verwahrt hatte.

Gauß' außerordentliche mathematische Begabung zeigte sich schon frühzeitig. Da er aus ärmlichen Verhältnissen stammte, gewährte ihm der Herzog von Braunschweig ab 1791 Stipendien für den Besuch des Gymnasiums und für das Göttinger Studium (1795–1798) sowie für die anschließende Zeit, bis Gauß im Jahre 1807 Mathematikprofessor und Direktor der Sternwarte in Göttingen wurde.

Es war durchaus nicht von Anfang an sicher, ob sich Gauß überhaupt ständig der Mathematik widmen sollte (seine wissenschaftliche Tätigkeit in diesem Fach hatte er bereits 1791 begonnen!). Denn er beherrschte mehrere Fremdsprachen und schwankte in der Wahl zwischen Mathematik und Philologie! Die Entscheidung fiel, als ihm der Nachweis gelungen war, daß das reguläre 17-Eck allein mit Zirkel und Lineal konstruierbar ist. Dieses geometrische Problem hatten seit dem Altertum Mathematiker zu lösen versucht und es schließlich für unlösbar erklärt. Gauß fand die Lösung 1796 quasi als Nebenergebnis bei zahlentheoretischen Untersuchungen.

Bereits 1799 promovierte Gauß, und zwar in einem für die damalige Zeit wohl denkwürdigen Verfahren: Die Universität Helmstedt verlieh dem Studenten der Göttinger Universität in Abwesenheit und ohne mündliche Prüfung den Doktorgrad für den ersten exakten Beweis des Fundamentalsatzes der Algebra. Schon 2 Jahre später veröffentlichte er seine umfangreichen *Disquisitiones aritmeticae,* mit deren Erscheinen er unter den Mathematikern weltberühmt wurde und die als Beginn der neueren Zahlentheorie gelten.

Neben weiteren zahlreichen, bis heute gültigen Neuschöpfungen auf verschiedenen Gebieten der reinen Mathematik (Fehlertheorie, Theorie der elliptischen Funktionen, Flächentheorie, Primzahltheorie, Darstellung der komplexen Zahlen, Potentialtheorie) vollbrachte Gauß auch bedeutende praxisorientierte Leistungen in anderen wissenschaftlichen Bereichen. So übernahm er 1818 einen Auftrag zur Vermessung des Königreichs Hannover – eine Arbeit die 25 Jahre andauerte und bei der insgesamt 2600 trigonometrische Punkte von der Nordsee bis zum Inselsberg eingemessen wurden. Für diese Arbeiten (an denen er 5 Jahre persönlich teilnahm) erfand er eigens das *Heliotrop*, ein Meßgerät, bei dem das Sonnenlicht für Vermessungssignale über große Entfernungen ausgenutzt wird. Mit der Vermessung des (damals größten vermessenen) Dreiecks zwischen dem Brocken (1142 m), dem Inselsberg (915 m) und dem Hohen Hagen (508 m) wollte Gauß auch nachprüfen, bis zu welchem Genauigkeitsgrad die Euklidische Geometrie in der realen Welt gilt.

In der Mechanik stellte er das *Prinzip des kleinsten Zwangs* auf, in der Optik verbesserte er die bis dahin üblichen Methoden der Strahlengangberechnung für ein Linsensystem. In der Astronomie befaßte er sich unter anderem mit Fragen der Zeitrechnung und des Kalenders. So hat er eine praktikable Regel zur Berechnung des Osterdatums im gregorianischen Kalender hergeleitet.

Im Jahre 1828 weilte Gauß als persönlicher Gast Alexander von Humboldts auf der *Versammlung Deutscher Naturforscher und Ärzte* in Berlin. Hier lernte er den Physiker Wilhelm Weber (1804–1891) kennen, mit dem er ab 1831 in Göttingen den Erdmagnetismus erforschte und die Zusammenhänge zwischen Magnetismus und Elektrizität untersuchte. Aus dieser gemeinsamen Tätigkeit resultierte 1832 die Erfindung des *Magnetometers* und 1833 die des ersten *elektromagnetischen Telegraphen*, durch den Webers physikalisches Institut mit der Sternwarte verbunden wurde. Gauß selbst äußerte danach, daß nun nichts anderes als technische und finanzielle Fragen zu lösen seien, um zu einem Nachrichtensystem über die ganze Erde zu gelangen.

Die Zusammenarbeit von Gauß und Weber brachte übrigens auch die wichtige Einführung des absoluten Maßsystems, also die Rückführung aller Maßeinheiten auf die drei Grundgrößen der Länge, Zeit und Masse.

Des weiteren waren Gauß und Weber engagiert in der von A. v. Humboldt gegründeten ersten wissenschaftlichen Gesellschaft, dem sogenannten Magnetischen Verein, tätig, für den erstmals nach standardisierten Verfahren und zu festgelegten Zeiten weltweit das Erdmagnetfeld gemessen wurde. Diese Aktivität des Magnetischen Vereins darf wohl als Vorläufer aller späteren internationalen Kooperationsunternehmen bis hin zum Geophysikalischen Jahr 1957/58 angesehen werden.

Bei der unglaublich großen Vielseitigkeit von Gauß nimmt es kaum noch wunder, daß ihm auch die Rechentechnik eine Reihe von Algorithmen verdankt, die vielfach zur Grundausstattung moderner Software gehören. Erwähnt seien in diesem Zusammenhang die *Gaußschen Quadraturverfahren* zur numerischen Berechnung ein- oder mehrdimensionaler Integrale, bei denen jeweils nur die Anzahl, nicht aber die Lage der Nullstellen vorgeschrieben wird. Häufig verwendet wird auch der *Gaußsche Algorithmus*, ein praktikables Eliminationsverfahren zur Auflösung linearer Gleichungssysteme, gleichermaßen geeignet zur Determinantenberechnung und zur Matrizeninversion. Für die Lösung ebensolcher Probleme dient das *Gauß-Seidelsche Iterationsverfahren*, das 1874 von Ph. L. Seidel angegeben wurde, aber schon von Gauß in verschiedenen Abwandlungen benutzt worden ist, wie sich nachträglich dank R. Dedekinds Ermittlungen (1901) herausgestellt hat. Große Bedeutung für die praktische Anwendung hat die auf Gauß zurückgehende *Ausgleichsrechnung* erlangt. Sie verfolgt das Ziel, aus fehlerhaften Meßwerten Näherungswerte für die zu messenden Größen zu gewinnen und deren Genauigkeit anzugeben. Benutzt wird dazu seine *Methode der kleinsten (Fehler-)Quadrate*, die Gauß zuerst in der Astronomie und Vermessungskunde eingeführt hat. Ihre erste weltweit beachtete Anwendung fand diese Methode 1801 durch Gauß selbst bei seiner Bahnberechnung des Planetoiden *CERES* aus nur ganz wenigen Positionsmessungen. Auf Grund seiner „zur Bewunderung genauen" Berechnung gelang den Astronomen die aufsehenerregende Wiederentdeckung dieses aus der Sicht entschwundenen Planeten. Noch heute werden bei den computergestützten Berechnungen von Umlaufbahnen die Gaußschen Methoden zugrunde gelegt!

Kein Geringerer als Felix Klein, der 1886–1923 ebenfalls an der Göttinger Universität als Mathematiker wirkte, hat zu Beginn einer seiner Vorlesungen Gauß mit folgenden Worten gewürdigt: „Wenn wir uns fragen, worin eigentlich das Ungewöhnliche, Einzigartige dieser Geisteskraft liegt, so muß die Antwort lauten: Es ist die Verbindung der größten Einzelleistung in jedem ergriffenen Gebiet mit größter Vielseitigkeit; es ist das vollkommene Gleichgewicht zwischen mathematischer Erfindungskraft, Strenge der Durchführung und praktischem Sinn für die Anwendung bis zur sorgfältig ausgeführten Beobachtung und Messung einschließlich; und endlich, es ist die Darbietung des großen selbstgeschaffenen Reichtums in der vollendetsten Form".

Dr. Klaus Biener

Einführung in Forth-83

Teil 1

Dr. Hartmut Pfüller (Leiter), Dr. Wolfgang Drewelow, Dr. Bernhard Lampe, Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik

Gliederung

1. Einführung und Kernwortschatz

Die dialogorientierte Programmiersprache Forth wurde in den sechziger Jahren von Charles H. Moore in den USA entwickelt und ab 1971 zunächst für die Echtzeitsteuerung von Radioteleskopen eingesetzt. Entwicklungsziele waren maximale Handlichkeit der Sprache zwecks hoher Produktivität des Programmierers und größte Einfachheit des Übersetzerkonzepts. Herausgekommen ist eine organische und kompakte Einheit von Sprache und Übersetzer. Forth ist erweiterbar, das heißt, der Quelltext kann für sich selbst die eigenen Ausdrucksmittel ändern und neue Ausdrucksmittel höheren Niveaus erzeugen. Damit ist problemnahe Programmierung für nahezu jede Anwendung in einheitlicher Softwareumgebung und *in einem Zuge* möglich, also ohne Mehrpaßübersetzung. In Richtung niederen Niveaus sind im Quelltext die Maschinenbefehle des Wirtsprozessors verwendbar. Das ist nützlich für direkte Gerätesteuerungen und Zugriffe auf Treiber.

Forthsysteme sind je nach Ausstattung etwa drei KByte bis über 16 KByte groß und für praktisch alle Rechner verfügbar. Forth wurde inzwischen in mehreren Etappen standardisiert; die vorliegende Beschreibung folgt dem Standard Forth-83. Eventuelle Unklarheiten beim Nachvollziehen der Beispiele sollten Veranlassung sein, die Verträglichkeit des benutzten Systems mit diesem Standard zu überprüfen.

1.1. Die Softwarekonzeption von Forth

1.1.1. Die Architektur von Forth

Das Forthsystem ist ein Rechenprogramm, das gleichzeitig als Betriebssystem, als Compiler und als Kommandoprozessor (*Textinterpreter*) arbeitet. Bild 1.1 soll das an einem Schichtenmodell anschaulich machen. Die niedrigste Schicht enthält alle Programmmoduln, die aus Maschinencode gebildet sind. Dieses Prinzip, die Gegebenheiten einer konkreten Hardware mittels eines zugeordneten Pakets von Maschinenprogrammen *abzufangen*, ist mit dem BIOS-Teil des Betriebssystems CP/M vergleichbar. Auch die nächsthöheren Schichten sind in Moduln gegliedert, allerdings bestehen diese nicht aus Maschinencode, sondern aus sogenanntem Forthcode. Das sind Listen von Adressen, wobei jede Adresse als Pointer zu einem bestimmten Modul aus Maschinencode oder aus Forthcode verweist. Die Betriebssystemschicht und die darüberliegenden sind dadurch maschinenunabhängig, also portabel. Alle Schichten können jederzeit auch im Dialog erweitert werden.

1.1.2. Das Wortkonzept

Das gesamte Forthsystem besteht aus einer größeren Anzahl (je nach Systemgröße z. B. etwa 70...300) von relativ kleinen Moduln (teils aus Maschinencode, teils aus Forthcode). Alle diese Grundfunktionen sind im Hauptspeicher lexikonförmig geordnet aufbewahrt. Wegen der tatsächlichen Ähnlichkeit mit einem Lexikon heißt dieser Programmteil *Wörterbuch*, und jeder der einzelnen Moduln wird als *Wort* bezeichnet. Die Worte im Wörterbuch dienen als Kommunikationsmittel zwischen Mensch und Rechner und müssen demzufolge von beiden *verstanden* werden. Zu diesem Zweck ist jeder Eintrag zweigeteilt und so aufgebaut, daß der Name des Wortes für den Menschen als Text lesbar und der zugehörige Codeteil für den Rechner ausführbar ist. Vier Beispiele für Namen sind:

; */MOD 2! VOCABULARY

Die Namensbildung ist sehr freizügig; Sonderzeichen sind an beliebiger Stelle erlaubt oder können auch einzeln als Name gelten.

1.1.3. Das Stapelkonzept

Außer Namen können im Eingabetext von Tastatur oder Massenspeicher natürlich auch Zahlen erscheinen. Diese Zahlen werden eine nach der anderen in das *interne Format* konvertiert, so zum *Parameterstapel* (Parameterstack) geschafft und dort abgelegt. Zur internen Wertdarstellung dienen durchgängig 16-Bit-Dualzahlen; für negative Zahlen wird die Zweierkomplementdarstellung verwendet. Auf dem Parameterstapel bleiben die Werte dann so lange liegen, bis sie von irgendeinem Wort wieder abgeholt (*verbraucht*) werden. Damit ist gleichzeitig angedeutet, wie Forthworte mit lokalen Eingangsparametern versorgt werden: Die Parameter müssen irgendwann vor Aktivierung des Wortes auf dem Parameterstapel hinterlegt worden sein. Falls mit mehreren Werten hantiert wird, gilt das LIFO-Prinzip (englisch: last in – first out). Entsprechend wird mit lokalen Ausgangsparametern verfahren: Falls ein Wort lokale Ergebniswerte liefert, bleiben diese nach Ausführung des Wortes auf dem Stapel liegen und sind so zum Beispiel wieder als Eingangsparameter für nachfolgende Worte verfügbar. Da auf diese Weise bei Aufrufen die aus anderen Programmiersprachen bekannten Listen von aktuellen und formalen Parametern entfallen, spricht man bei Forth auch von *impliziter Parameterübergabe*. Für Programmdokumentationen oder Quelltextkommentare kann es erforderlich sein, genauere Angaben über Anzahl und Art der lokalen Ein- und Ausgangsparameter zu machen. Dafür hat sich in Forth eine bestimmte Art der Kommentierung eingebürgert: In runden Kommentarklammern wird notiert, wie der Stapel vor und nach Ausführung des kommentierten Wortes belegt ist. Die Notation

(n1 n2 n3 ===> n4 n5)

bedeutet in diesem Sinne, daß das kommentierte Forthwort drei Eingangsparameter vom Stapel holt – mit n3 an der Stapelspitze – und zwei neue Werte – mit n5 an der Stapelspitze – als Ausgangsparameter hinterläßt. Der Pfeil symbolisiert die Ausführung des Wortes.

1.1.4. Das Erweiterungskonzept

Die Vorgehensweise beim Programmieren in Forth soll an einem Steuerprogramm für einen hypothetischen x-y-Schreiber erläutert werden. Dieser Plotter sei so einfach, daß seine Hardware nur drei Funktionen kennt, und zwar

a) *Stift anheben*
b) *Stift absenken*
c) *geradenwegs die Absolutposition (x, y) anfahren.*

Wie diese Funktionen von der Rechnerperipherie aus zu aktivieren sind, soll bekannt sein. Programmieren in Forth heißt nun, das Forthsystem zu erweitern, indem der Programmierer zu den im Wörterbuch schon existierenden Worten neue Worte *hinzudefiniert*, nämlich solche, die Schritt für Schritt die Programmieraufgabe lösen. Dabei können alle schon im Wörterbuch existierenden Worte ausgenutzt werden. Zum Definieren neuer Worte gibt es (natürlich *auch* im Wörterbuch!) *Definitionsworte*. Ein neues Wort für die Maschinencodeschicht wird zum Beispiel durch Voranstellen des Definitionswortes **CODE** vor einen selbstzuwählenden Namen erzeugt. Mit dem Wort **END-CODE** wird ein solches Maschinenprogramm dann wieder beendet. Der Forthprogrammierer geht nun üblicherweise so vor, daß er entsprechend den Gerätedaten das Assemblerprogramm konzipiert und dann eintippt:

```
CODE HEBEN  . . . . . . . . . .  END-CODE
CODE SENKEN . . . . . . . . . .  END-CODE
```

Da die konkreten Assemblerbefehle hier nicht im Mittelpunkt stehen, wurden sie nur

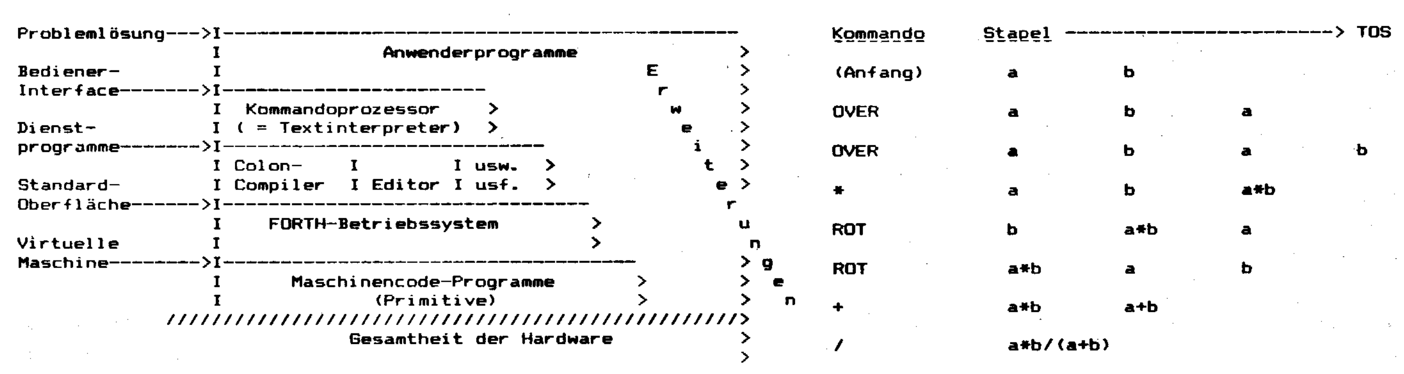


Bild 1.1 Schichtenmodell eines Forthsystems

Kommando	Stapel			---> TOS
(Anfang)	a	b		
OVER	a	b	a	
OVER	a	b	a	b
*	a	b	a*b	
ROT	b	a*b	a	
ROT	a*b	a	b	
+	a*b	a+b		
/	a*b/(a+b)			

Bild 1.5 Stapelbelegung während der Formelberechnung

```
2233  -77  444  (cr) ok          (wird auf Stapel gelegt)
.  (cr) 444 ok                   (letzter Wert entnommen)
.  .  (cr) -77 2233 ok           (nächste Werte entnommen)
```

Bild 1.2 Ausgabe von Zahlen mittels Punktbefehl

```
1  65535  -2  (cr) ok
U.  U.  U.  (cr) 65534 65535 1 ok
```

Bild 1.3 Vorzeichenlose Ausgabe von Zahlen

```
0  1  D.  (cr) 65536 ok
1  0  D.  (cr) 1 ok
```

Bild 1.4 Ausgabe von doppeltgenauen Zahlen

durch Punkte angedeutet. Nachdem nun die neuen Worte **HEBEN** und **SENKEN** definiert sind, kann der Programmierer durch deren Aufruf bei angeschlossenem Plotter kontrollieren, inwieweit die von ihm beabsichtigte Funktion korrekt ausgeführt wird. Die *Ausführung* von Worten erreicht man einfach durch Eintippen ihres Namens und Bestätigung der Eingabe mit der <Enter>-Taste. Diese *normale* Betriebsart von Forth heißt deshalb auch *Ausführungsmodus*. Wenn die neuen Worte augenscheinlich nicht korrekt arbeiten, müssen sie berichtigt neu eingegeben werden, bis der gewünschte Erfolg erreicht ist. Danach werde in ähnlicher Weise die dritte Funktion als Maschinenprogramm codiert:

CODE ANFAHREN . . . . END-CODE

Der Programmierer hat **ANFAHREN** als sinnfälligen Namen für diese Funktion gewählt. Hier ist nun zu beachten, daß dieses Wort die Parameter x und y für die Zielkoordinaten benötigt. In Quellprogrammen gehört es zu gutem Forthstil, mit dem neudefinierten Namen entsprechende Hinweise als Kommentar in runden Klammern zu notieren.

CODE ANFAHREN (x y ===>)
. . . . END-CODE

Durch die Ausführung des Wortes **ANFAHREN** werden also zwei Werte vom Stapel entfernt (verbraucht). Auch die Funktion dieses Wortes kann nun im Dialog getestet werden, z. B. durch Eintippen von:

50 70 ANFAHREN

Wenn das Maschinenprogramm **ANFAHREN** korrekt ist, muß damit die Position (x = 50, y = 70) angefahren werden. Natürlich können Worte auch im Verbund arbeiten, zum Beispiel so:

HEBEN 0 0 ANFAHREN
SENKEN 100 50 ANFAHREN HEBEN

Nach Eingabe dieser Zeilen muß der Plotter eine Strecke vom Punkt (x = 0, y = 0) zum Punkt (x = 100, y = 50) zeichnen. Neben dem Definitionswort **CODE** für das Eintragen von Maschinencodeworten ins Wörterbuch gibt es weitere Definitionsworte. Das vielleicht wichtigste von ihnen ist der *Doppelpunkt*. Er dient zur Erweiterung der höheren Schichten um Moduln in Forthcode. Für den Plotter kann zum Beispiel ein Wort definiert werden, dessen Funktion es ist, zwei Punkte durch eine Linie zu verbinden:

: VERBINDEN (x2 y2 x1 y1 ===>)
ANFAHREN SENKEN ANFAHREN HEBEN ;

Hier werden die Worte **ANFAHREN**, **SENKEN** und **HEBEN** nach Eingabe *nicht* ausgeführt, sondern als Programm in Forthcode unter dem neuen *Stichwort* **VERBINDEN** ins Wörterbuch kompiliert. Das wird dadurch erreicht, daß der Doppelpunkt generell nach seiner Aktivierung die Betriebsart vom Ausführungsmodus in den sogenannten *Kompilationsmodus* umschaltet. Das abschließende Semikolon (*auch* ein Forthwort!) schaltet dann wieder vom Kompilationsmodus in den Ausführungsmodus zurück. Die Gesamtkonstruktion einschließlich Doppelpunkt und Semikolon wird Doppelpunktdefinition genannt. Nach der Definition kann natürlich auch das Wort **VERBINDEN** im Dialog getestet werden, zum Beispiel so:

80 100 0 20 VERBINDEN

Die Ausführung des Wortes **VERBINDEN** bedeutet nun, daß genau diejenigen Befehle ausgeführt werden, die in der Doppelpunktdefinition notiert sind, also: **ANFAHREN SENKEN ANFAHREN HEBEN**. Man mache sich klar, daß die Richtung vom Punkt (x = 0, y = 20) zum Punkt (x = 80, y = 100) führt: Die Koordinaten des Punktes (x = 0, y = 20) liegen obenauf und werden deshalb als erste vorgefunden und angefahren. In ähnlicher Weise können im Dialogbetrieb nach und nach weitere Worte definiert, schrittweise zu komplexeren Funktionen zusammengefaßt und getestet werden. Zum Beispiel könnte man die Worte definieren, die lediglich die Parameter von markanten Punkten des Achsenkreuzes liefern:

: NULLPUNKT (===> x0 y0) 0 0 ;
: XMAX (===> xmax y0) 100 0 ;
: YMAX (===> x0 ymax) 0 100 ;

Das erlaubt dann sinnfällige Wortsequenzen wie

. . . NULLPUNKT ANFAHREN . . .

Weiter könnte damit nun beispielsweise ein Wort zum Zeichnen der positiven Achsen definiert werden:

: ACHSEN (===>)
NULLPUNKT YMAX VERBINDEN
XMAX NULLPUNKT VERBINDEN ;

Wenn Programme ausprobiert werden sollen, ohne daß das anzusprechende Gerät verfügbar ist, kann man sich dadurch helfen, daß die Grundfunktionen durch Doppelpunktworte definiert werden. Dabei muß das Stapelverhalten korrekt nachgebildet werden. Anstatt die eigentliche Funktion auszuführen, können zum Beispiel Zeichenkettentexte auf dem Bildschirm angezeigt werden:

: HEBEN ." heben " ;
: SENKEN ." senken " ;
: ANFAHREN (x y ===>)
. . ." anfahren " ;

Nach dem Eintippen dieser Worte könnten die obigen Beispiele zur Plottersteuerung nachvollzogen werden.

1.2. Das Bedienerinterface

1.2.1. Integerzahlen

Der Textinterpreter von Forth versucht für jeden (in Leerzeichen eingegrenzten) Eintrag zunächst, die Zeichenfolge als Name eines Forthwortes im Wörterbuch aufzufinden. Falls das mißlingt, wird als nächstes geprüft, ob die eingebene Zeichenfolge als Zahl verstanden werden kann, das heißt, ob sie ausschließlich aus gültigen Ziffernzeichen (allenfalls mit führendem Minuszeichen) besteht. Wenn diese Prüfung erfolgreich ist, wird die Ziffernzeichenfolge in die rechnerinterne Zahlenwertdarstellung umgewandelt und so auf dem Parameterstapel abgelegt.

1.2.2. Worte

Jedes eingegebene Wort, ob es nun eine (ein- oder mehrstellige) Zahl oder ein im Wörterbuch enthaltenes Forthwort repräsentiert, muß vom nachfolgenden Wort im Programmtext durch mindestens ein Leerzeichen getrennt sein. Nachfolgend wird die Benutzung von Zahlen und Forthworten am Beispiel einiger Ausgabebefehle demonstriert.

Wortname: .
Stapeleffekt: (n ===>)
Funktion: Das ASCII-Zeichen *Punkt* ist in Forth der einfache Ausgabebefehl; er holt (d. h. entfernt) den obersten 16-Bit-Wert vom

16-Bit-Wert vom Parameterstapel, faßt ihn als vorzeichenbehaftete Zahl in Zweierkomplementdarstellung (−32768 ... 32767) auf, wandelt ihn in die externe Ziffernzeichendarstellung und sendet diese ASCII-Zeichenkette als externe Darstellung der Dualzahl zum Ausgabegerät (Bild 1.2).
Zur Unterscheidung von den Bedienereingaben ist die Rechnerreaktion in diesen Dialogmitschnitten jeweils durch Unterstreichung gekennzeichnet. Das zusätzlich eingefügte (cr) soll bedeuten, daß vor dieser Stelle die Entertaste betätigt wurde.
Wortname: **U.**
Stapeleffekt: (u ===>)
Funktion: Ausgabebefehl wie Punkt; faßt den Wert als vorzeichenlose Dualzahl (0 ... 65536) auf (Bild 1.3).
Wortname: **D.**
Stapeleffekt: (*n n* ===>) oder identisch: (d ===>)
Funktion: Ausgabebefehl; holt die oberen beiden (16-Bit-)Werte vom Stapel und faßt dieselben gemeinsam als 32-Bit-Ganzzahl in vorzeichenbehafteter Zweierkomplementdarstellung (−2147483648 bis 2147483647) auf (Bild 1.4).
Eine Kurzbeschreibung aller aufgeführten Befehle ist in Tafel 1.2 zu finden. Da diese in vielen Fällen ausreichend ist, wird es im weiteren Verlauf seltener erforderlich sein, Worte so ausführlich wie eben vorgeführt zu beschreiben. In allen drei obigen Beschreibungen wurde absichtlich dazugesagt, wie der Ausgabebefehl den Stapelwert *auffaßt*. Diese *Auffassungsfrage* gilt nicht nur für Ausgabebefehle, sondern generell: Auf dem Stapel werden physisch nur 16-Bit-Werte verwaltet. Der Programmierer allein entscheidet bei neuen Definitionen durch die von ihm organisierte Weiterverwendung darüber, ob sein Forthwort einen Wert als vorzeichenbehaftet, als ASCII-Zeichen mit nur sieben gültigen Bits oder mit weiteren Parametern gemeinsam als mehrfachgenauen Wert benutzt oder auf völlig andere Weise. Bei der Verwendung von bereits existierenden Worten muß er sich natürlich darüber informieren, welche Bedeutung diese jeweils den Parametern beilegen.

1.3. Der Parameterstapel

Da die Werte auf dem Parameterstapel für sehr verschiedene Anwendungsfälle als Eingangsparameter verwendet werden können, ist es manchmal zweckmäßig, vorhandene Werte zu kopieren oder umzusortieren. Dafür sind die in den nachfolgenden Unterabschnitten aufgeführten Worte nützlich.

1.3.1. Vervielfältigung
Die sechs Worte
DUP OVER PICK ?DUP 2DUP 2OVER
stellen alle in irgendeiner Weise Kopien von bestimmten Werten irgendwo auf dem Stapel her, die dann als Ergebnisparameter auf das obere Stapelende (englisch: Top of stack, Abk.: TOS) abgelegt werden. Die Stapeldiagramme und Kurzbeschreibungen sind in Tafel 1.2 enthalten. Mit **PICK** kann ein beliebiger Wert zum TOS kopiert werden, dessen Position man allerdings explizit angeben muß. **?DUP** dupliziert den Wert im TOS genau dann, wenn er nicht gleich Null ist. Das ist besonders in Verbindung mit Entscheidungsstrukturen nützlich.

1.3.2. Ordnungsbefehle
Jedes der fünf Worte
ROT SWAP ROLL 2ROT 2SWAP
leistet in irgendeiner Weise eine Umsortierung von Werten auf dem Parameterstapel; der Füllstand des Stapels wird dabei nicht geändert.

1.3.3. Entfernen von Werten
Die Worte **DROP** und **2DROP** entfernen vom TOS einen 16-Bit-Wert bzw. einen 32-Bit-Wert. Das wird zum Beispiel dann benötigt, wenn erst nach Entscheidungsfolgen klar wird, welcher von mehreren Parametern weiter benötigt wird und welcher nicht. Überflüssige Werte können dann durch Umsortierung nach oben gebracht und mit diesen Befehlen *vernichtet* werden.

1.3.4. Stapeltiefe
Das Wort **DEPTH** (deutsch: Tiefe) liefert als Ergebnisparameter eine Zahl, die angibt, mit wieviel 16-Bit-Werten der Parameterstapel gefüllt ist.

1.4. Arithmetische Operationen

1.4.1. Umgekehrte polnische Notation
Im Plotterbeispiel wurde eine Befehlsfolge **NULLPUNKT ANFAHREN** benutzt. Dabei folgt auf den (zusammengesetzten) *Parameter* **NULLPUNKT** die *Operation* **ANFAHREN**. Nach diesem Prinzip kann auch mit Zahlen operiert werden. Zum Beispiel gibt es in Forth eine Operation für das Addieren; das entsprechende Forthwort hat den Namen + erhalten. Angenommen, auf dem Stapel liege ein Wert **5**, dann führt die Befehlsfolge **3 +** dazu, daß die **5** um den Wert **3** erhöht, also zu einer **8** wird. Wesentlich an dieser Betrachtung ist, daß in Forth generell die Operation *hinter* dem (oder den) Operanden steht (sogenannte Postfixnotation; andere übliche Bezeichnungen dafür sind UPN = umgekehrte polnische Notation oder englisch: RPN = reverse Polish Notation). In Verbindung mit der LIFO-Verwaltung des Parameterstapels wird sich das als sehr praktisch erweisen. Trotzdem ist es etwas unüblich, denn die in der Schule gelehrte Notation setzt den Operator *zwischen* die Operanden (sogenannte Infixnotation). Es läßt sich zeigen, daß daneben auch noch eine Präfixnotation möglich ist (Operator *vor* den Operanden; z. B. von der Programmiersprache LISP bevorzugt) und daß alle diese Notationen ineinander überführbar sind. Ungewohnt ist die Postfixnotation am Anfang vielleicht am ehesten bei größeren Formeln.

1.4.2. Die Grundrechenarten
Tafel 1.2 zeigt die Funktion der sechs Worte
+ − * / D+ D−
Die ersten vier erwarten jeweils zwei Parameter auf dem Stapel, entfernen diese und hinterlassen im TOS das Ergebnis der Operation. Der Programmierer muß wissen, daß Bereichsüberschreitungen zum Beispiel bei Addition oder Multiplikation nicht reklamiert werden. Weiter ist wichtig zu beachten, daß die Division nur den ganzzahligen Teil des Quotienten liefert; ein eventueller Rest wird ignoriert. Alternativen dazu enthält der nächste Abschnitt. Die beiden Worte **D+** und **D−** hinterlassen entsprechend die doppeltgenaue Summe bzw. Differenz von doppeltgenauen Eingangswerten. Die praktische Benutzbarkeit der Postfixnotation in Verbindung mit Kopier- und Ordnungsbefehlen soll an einem Beispiel gezeigt werden. Angenommen, auf dem Stapel liegen zwei Parameter a und b, und es soll der Ausdruck
(a * b)/(a + b) berechnet werden. Das ist mit der folgenden Sequenz möglich:
OVER OVER * ROT ROT + /
Da das nicht besonders anschaulich ist, kann es für Entwicklungszwecke zum Selbstverständnis nützlich sein, in einer Art *vertikalen Notation* die aktuelle Belegung des Stapels nach jedem Kommando als Kommentar festzuhalten (Bild 1.5).

1.4.3. Division mit Rest
Die beiden Worte
/MOD MOD
können benutzt werden, wenn die Vernachlässigung des Restes bei der ganzzahligen Division nicht zulässig ist. **/MOD** liefert dann im TOS den Quotienten und darunter den Rest. **MOD** liefert nur den Rest selbst.

1.4.4. Skalierung
Die beiden Worte
***/ */MOD**
erwarten drei Werte x y z als Parameter und berechnen den Wert von x*y/z, wobei die Zwischenergebnisse intern mit doppelter Genauigkeit geführt werden. */ liefert einen Ergebnisparameter. Mit diesem Operator läßt sich unsere Formel (a*b)/(a+b) aus Abschnitt 1.4.2. auch so berechnen:
OVER OVER + */
Ganz ähnlich wie der eben besprochene Befehl */ liefert */MOD ebenfalls das Ergebnis im TOS, dazu aber als zweiten Wert (unter dem TOS) zusätzlich den Rest der abschließenden Division.

1.4.5. Gemischtgenaue Operationen
Gemischtgenaue Operationen sind solche, bei denen Parameter verschiedener Genauigkeit eine Rolle spielen. Zum Beispiel erwartet **UM*** zwei einfachgenaueParameter (vorzeichenlos) und hinterläßt deren doppeltgenaues Produkt (ebenfalls vorzeichenlos). **UM/MOD** erwartet im TOS einen einfachgenauen Divisor und darunter einen doppeltgenauen Dividenden. Das Ergebnis ist wieder einfachgenau; alle Parameter werden als vorzeichenlos aufgefaßt.

1.4.6. Begrenzerbefehle
Die vier Befehle
MAX MIN DMAX DMIN
erwarten je zwei Eingangsparameter und liefern als Ausgangsparameter das Extremum der beiden.

1.4.7. Vorzeichenbefehle
ABS und **DABS** bilden den Betrag von einfach- bzw. doppeltgenauen Zahlen; **NEGATE** und **DENEGATE** bilden jeweils das Zweierkomplement des einfach- bzw. doppeltgenauen Eingangsparameters. Damit wird eine Vorzeichenumkehr erreicht.

1.4.8. Maschinenahe Operationen

Die Gruppe

1+ 1− 2+ 2− 2/ D2/

faßt solche Operationen zusammen, die in Programmen erfahrungsgemäß häufig vorkommen und für die gleichzeitig besonders schnelle Realisierungen in Maschinencode möglich sind. Das ist das Erhöhen und das Vermindern um eins und um zwei sowie eine bitweise Rechtsverschiebung des Eingangsparameters (einfach- bzw. doppeltgenau), die in der internen Darstellung eine Division durch zwei realisiert. *wird fortgesetzt*

***Tafel 1.1* Symbole für die Bedeutung der Stapelparameter**

TOS	oberster Wert auf dem Parameterstapel (englisch: Top Of Stack)
+n	Stapeleintrag im Bereich 0<x<=32767
16b	Stapeleintrag mit 16 gültigen Bits
32b	doppeltgenauer Stapeleintrag mit 32 gültigen Bits
8b	Stapeleintrag mit 8 gültigen Bits (b0/ bis b7)
?	Stapeleintrag mit den Werten "true" (? ≠ 0) bzw. "false" (? = 0)
addr	Stapeleintrag, der als Adresse angesehen wird
c	Stapeleintrag, der ein (ASCII-)Zeichen spezifiziert
d	doppelter Stapeleintrag im Bereich −2147483648<x<=2147483647
+d	positive doppelt genaue Zahl 0<x<=4294967295
n	Stapeleintrag im Bereich −32768<x<=32767
u	Stapeleintrag im Bereich 0<x<=65535
ud	doppelter Stapeleintrag im Bereich 0<x<=4294967295
w	Stapeleintrag im Bereich −32768<x<=65535
wb	Stapeleintrag im Bereich −128<x<=255
wd	Stapeleintrag im Bereich −2147483648<x<=4294967295

Zu diesem Kurs

Hier wird dem interessierten Leser die Möglichkeit gegeben, sich über einige der wesentlichen Eigenarten und Potenzen von Forth grob zu orientieren. Wer dadurch angeregt wird, sich ernsthafter mit diesem zukunftsträchtigen Softwarekonzept zu befassen, dem sei das sorgfältige Studium der Bücher von Brodie /1/, /2/ und Zech /3/ empfohlen. Daneben kann man sich in der Spezialliteratur über Themen informieren, die in diesem Kurs nur kurz oder gar nicht erwähnt werden, zum Beispiel: Editieren in Forth / Assemblerprogrammierung / Interrupts, Echtzeit und Multitasking / Metakompilation und Crosscompilation / Fließkommarechnung / Forthprozessor in Hardware.

Hilfreich ist sicherlich auch der Kontakt zu anderen Forthprogrammierern. Für beruflich Interessierte bietet sich hier die Kammer der Technik an. Dort arbeitet in der wissenschaftlichen Sektion Computer- und Mikroprozessortechnik des Fachverbandes Elektrotechnik ein Fachausschuß Forth. Auch außerberuflich an Forth Interessierte haben sich beim Kulturbund in Leipzig zu einer Interessengemeinschaft Forth zusammengefunden, die landesweit aktiv ist.

***Tafel 1.2* Kurzbeschreibung der Forthworte**

Name	Stapeleffekt	Beschreibung
Zahlenausgaben		
.	(n ===>)	16-Bit-Wert mit MSB als Vorzeichen ausgeben
U.	(u ===>)	16-Bit-Wert als Positivwert ausgeben
D.	(d ===>)	32-Bit-Wert mit MSB als Vorzeichen ausgeben
Parameterstapel; Vervielfältigungen		
DUP	(16b ===> 16b 16b)	obersten Stapelwert identisch duplizieren
OVER	(16b0 16b1 ===> 16b0 16b1 16b0)	zweiten Wert zum TOS kopieren
PICK	(16bj 16bi 16bh ... 16b0 i ===> 16bj 16bi 16bh ... 16b0 16bi)	i-ten Wert zum TOS kopieren
?DUP	(0 ===> 0) (16b ===> 16b 16b)	TOS nur dann duplizieren, wenn er nicht null ist
2DUP	(32b ===> 32b 32b)	32-Bit Wert duplizieren
2OVER	(32b0 32b1 ===> 32b0 32b1 32b0)	zweiten doppeltgenauen Wert zum TOS kopieren
Ordnungsbefehle		
ROT	(16b0 16b1 16b2 ===> 16b1 16b2 16b0	dritten Wert nach oben "rollen"
SWAP	(16b0 16b1 ===> 16b1 16b0)	obere beide Werte tauschen
ROLL	(16bj 16bi 16bh ... 16b0 i ===> 16bj 16bh ... 16b0 16bi)	i-ten Wert nach oben "rollen"
2ROT	(32b0 32b1 32b2) ===> 32b1 32b2 32b0)	dritten doppeltgenauen Wert nach oben "rollen"
2SWAP	(32b0 32b1 ===> 32b1 32b0)	obere beide doppeltgenauen Werte tauschen
Entfernen von Werten		
DROP	(16b ===>)	TOS entfernen
2DROP	(32b ===>)	doppeltgenauen Wert vom TOS entfernen
Stapeltiefe		
DEPTH	(===> +n)	Gesamtzahl der 16-Bit-Werte auf dem Stapel
Grundrechenarten		
+	(w1 w2 ===> w3)	liefert w3 als Summe aus w1 und w2
−	(w1 w2 ===> w3)	liefert w3 als Diffrenz aus w1 − w2
*	(w1 w2 ===> w3)	bildet w3 als Produkt aus w1 und w2
/	(n1 n2 ===> n3)	bildet n3 als Quotient n1/n2
D+	(wd1 wd2 ===> wd3)	addiert doppeltgenaue Werte wd1 und wd2 zu wd3
D−	(wd1 wd2 ===> wd3)	subtrahiert zwei doppeltgenaue Zahlen zu wd3
Division mit Rest		
/MOD	(n1 n2 ===> n3 n4)	bildet Quotient n4 und Rest n3 von n1/n2
MOD	(n1 n2 ===> n3)	bildet den Rest n3 der Division n1/n2
Skalierung		
*/	(n1 n2 n3 ===> n4)	n4 = n1*n2/n3; Zwischenergebnis n1*n2 ist doppeltgenau
*/MOD	(n1 n2 n3 ===> n4 n5)	n5 = n1*n2/n3; n4 ist der Rest bei der Division
Gemischtgenaue Operationen		
UM*	(u1 u2 ===> ud)	doppeltgenaues Produkt einfachgenauer Positivwerte
UM/MOD	(ud u1 ===> u2 u3)	Quotient u3 und Rest u2 von ud/u1
Vergleichende Befehle		
MAX	(n1 n2 ===> n3)	n3 ist die größere der beiden Zahlen n1 und n2
MIN	(n1 n2 ===> n3)	n3 ist die kleinere der beiden Zahlen n1 und n2
DMAX	(d1 d2 ===> d3)	d3 ist die größere der doppeltgenauen Zahlen d1 und d2
DMIN	(d1 d2 ===> d3)	d3 ist die kleinere der doppeltgenauen Zahlen d1 und d2
Vorzeichenbefehle		
ABS	(n ===> u)	u ist der Absolutbetrag von n
DABS	(d ===> ud)	ud ist der Absolutbetrag von d
NEGATE	(n1 ===> n2)	n2 ist das Zweierkomplement von n1
DNEGATE	(d1 ===> d2)	d2 ist das Zweierkomplement von d1
Maschinennahe Operationen		
1+	(w1 ===> w2)	w1 wird inkrementiert zu w2
1−	(w1 ===> w2)	w1 wird dekrementiert zu w2
2+	(w1 ===> w2)	w1 wird um 2 inkrementiert zu w2
2−	(w1 ===> w2)	w1 wird um 2 dekrementiert zu w2
2/	(w1 ===> w2)	w1 wird um ein Bit arithmetisch rechtsverschoben
D2/	(wd1 ===> wd2)	analog 2/ für doppeltgenaue Zahlen

Literatur

/1/ Brodie, L.: Programmieren in Forth. München: Carl Hanser Verlag 1984

/2/ Brodie, L.: In Forth Denken. München: Carl Hanser Verlag 1986

/3/ Zech, R.: Forth-83. München: Franzis Verlag 1987

✉ KONTAKT ☏

WPU Rostock, Sektion Technische Elektronik, WB Automatische Steuerungen, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 452 14

Entwurf von Gate-Array-Schaltkreisen Teil 1

Prof. Dr. Dietmar Müller
Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik

Allgemeiner Entwurfsablauf

Im Bild 1 sind mögliche Ebenen beim Entwurf von Schaltkreisen dargestellt.
Der angegebene Ablauf entspricht einer Strategie *Von oben nach unten* (top down). Dieses – im „VLSI-Zeitalter" anzustrebende – Vorgehen muß jedoch durch Aspekte des *Von unten nach oben* (bottom up) ergänzt und unterstützt werden. So muß – speziell beim GA-Entwurf – Wissen über die nutzbaren Funktionselemente der Funktionselemente-Bibliothek vorhanden sein, damit keine nichtrealisierbaren Teilstrukturen entworfen werden. Die Dialektik zwischen Top-down- und Bottom-up-Vorgehen ist ausführlich in /1/ dargestellt.
Der Top-down-Entwurf komplexer Systeme kann als eine mehrstufige Transformation der im Pflichtenheft formulierten und geforderten Funktion in eine – letztlich – geometrische Struktur verstanden werden. Bei nicht zu kleinen Aufgabenstellungen ist ein Top-down-Entwurf unter Ausnutzung allgemeiner Regeln und Vorgehensweisen, wie Nutzung des Hierarchie- und des Blockkonzeptes /1/ oder des Anstrebens von Wiederholstrukturen, des Parallelisierens, des Serialisierens u. a., sinnvoll. Dabei kommt es in wachsendem Maße darauf an, solche Methoden und Programme zu entwickeln, die der Anwender nutzen kann, ohne daß Detailkenntnisse vorhanden sind. Unter diesem Aspekt wird in schnell wachsendem Umfang auch auf der **Systemebene** die Rechnerunterstützung allgemein und außerhalb des Problemkreises GA-Entwurf perfektioniert. Dies geschieht durch das Entwickeln und Nutzbarmachen von immer komfortableren und leistungsfähigeren Entwurfs- und Beschreibungsmitteln und den dazugehörigen Simulationsprogrammen.

Werden die auf der Systemebene entworfenen und (teilweise oder vollständig) simulierten Teilkomplexe partioniert, so ergibt sich die **Register-Transfer-Ebene**. Diese ist durch den schrittweisen Transfer von Informationen zwischen Speicherplätzen – z. B. Registern – charakterisiert, wobei Informationen verändert bzw. mit anderen Informationen verknüpft werden können.

Es folgt die **Logikebene**. Hier werden die in der Register-Transfer-Ebene noch vorkommenden komplexen Schaltungen durch Strukturen auf Gatter-Niveau aufgelöst. Dieser Gatterentwurf wird ebenfalls durch zahlreiche strukturelle und funktionelle Beschreibungssprachen unterstützt, wobei eine große Zahl dieser Sprachen als Eingabemedium für die Simulatoren auf diesem Niveau dienen. Neben Simulatoren, die nur zwei logische Zustände bearbeiten, gibt es zahlreiche, deren Simulationsergebnisse sehr viel aussagekräftiger und genauer sind, da mehr als zwei Zustände unterschieden werden. Neben dieser statischen Simulation werden auch die zeitlichen Vorgänge bis zum Erreichen der statischen Endzustände durch

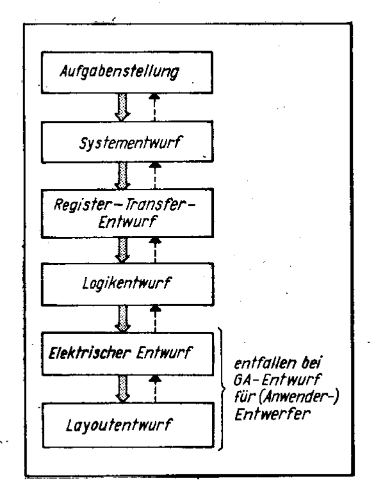


Bild 1 Allgemeine Entwurfsebenen

dynamische Logiksimulation – besser zeitliche Logiksimulation untersucht.
Diese oberen Entwurfsebenen bleiben die „Domäne" des Entwerfers, der durch seine konzeptionelle Arbeit, durch seine Idee, trotz aller Rechnerunterstützung maßgeblich über den Erfolg des Schaltkreises entscheidet. Dabei wird diese Kreativität bei VLSI-Entwürfen nicht darauf gerichtet sein, originelle Schaltungsdetails und transistorsparende Teilschaltungen zu entwerfen, sondern auf der Basis bewährter LSI-Komplexe und regulärer Wiederholstrukturen in kurzer Zeit funktionierende und prüfbare Schaltkreise zu entwerfen. Diese Entwurfsvorgänge von Routinearbeit – auch von Schreib- und Bedienarbeit am Rechner – zu befreien, ist ein weiterer Aspekt für das Entstehen der Beschreibungssprachen, deren Vielzahl ein Indiz für das Fehlen einer unter vielen Aspekten optimalen Sprache ist.

An den Logikentwurf schließt sich im allgemeinen der Entwurf in der Elektrik-Ebene an. Dieser beinhaltet den Ersatz der Gatterstrukturen durch Transistorschaltungen. Dieser elektrische Entwurf ist bei nichttrivialen Transistorzahlen manuell nicht beherrschbar, so daß sehr frühzeitig gerade für dieses Niveau eine Rechnerunterstützung erarbeitet wurde. Die mathematisch sehr anspruchsvollen und rechentechnisch aufwendigen Algorithmen zur Netzwerkanalyse sind zum Beispiel in /2/ und /3/ übersichtlich dargestellt und erfahren ständig Verbesserungen und Erweiterungen.
Der Entwurfsprozeß endet mit der Umsetzung des berechneten elektrischen Verhaltens in adäquate geometrische Strukturen, das heißt dem Entwerfen des Layouts der Masken für die einzelnen technologischen Schritte. Auch hier ist der Mensch durch die Fülle der Daten völlig überfordert, so daß nur in der Anfangszeit das Layout manuell entworfen wurde; heute existiert auch für diesen Entwurfsschritt eine gediegene und teilweise vollständige Rechnerunterstützung. Der Layoutentwurf führt zu Datenträgern mit den Steuerdaten für die Maskenherstellung oder für eine eventuelle Direktbelichtung mittels Elektronenstrahl. Neben einer Vielzahl von Problemen gestattet gerade diese Art ein Höchstmaß an Flexibilität bei der Schaltkreisherstellung (z. B. Multi-Projekt-Chips) und an Zeitgewinn.
Abschließend soll noch auf eine wesentliche Aufgabe hingewiesen werden, deren Lösung eigentlich mit dem Systementwurf beginnen muß. Es ist dies die zu entwerfende Prüf- und Meßbarkeit des Schaltkreises. Während dieser prüfgerechte Entwurf in jedem Falle erstrebenswert ist, *muß* er beim ASIC-Entwurf realisiert werden.

Entwurfsablauf bei Gate-Array-Schaltkreisen

Ist entsprechend Bild 1 der Systementwurf so weit detailiert, daß Teilkomplexe als Gate-Array-Schaltkreis realisiert werden können, oder ist die Aufgabenstellung so übersehbar, daß ein Gate-Array-Schaltkreis ohne „Systementwurf" entstehen kann, so muß für diesen ein Logikplan entworfen werden. Dies bedeutet, die funktionelle Beschreibung der Aufgabenstellung in Form *Boole*scher Gleichungen, Graphen oder anderer bekannter Beschreibungsformen in eine Struktur zu überführen, die die geforderte Funktion realisiert. Beim Gate-Array-Entwurf hat dies, wie bereits bemerkt, durch ausschließliche Nutzung der in der Elementebibliothek enthaltenen und damit für den Nutzer verfügbaren Funktionselementen zu geschehen. Mit diesen Funktionselementen und wegen der Notwendigkeit eines prüfgerechten Entwurfs sind sequentielle Schaltungen nicht in beliebiger Form realisierbar (siehe Teil 2).
Der Entwurfsablauf bei Gate-Array-Schaltkreisen ist im Bild 2 skizziert.
Im Rahmen der Masterkenngrößen (Anzahl potentieller Funktionselemente der Elementebibliothek wird der (Anfangs-)Logikplan mittels einer Beschreibungssprache oder bei Vorhandensein von Grafikeditoren die strukturelle Verschaltung der Funktionslemente in den Rechner eingegeben.
Mit dieser Eingabe ist die Simulation der Funktion bis zur vollständigen Verifikation möglich. In dieser Phase – der **Entwurfssimulation** – praktiziert der kreative Entwerfer, durch leistungsfähige Simulationsprogramme unterstützt, echten rechnerunterstützten Entwurf. Durch Dialogorientierung der Simulatoren erhält der Entwerfer nach kurzer Reaktionszeit des Rechners die

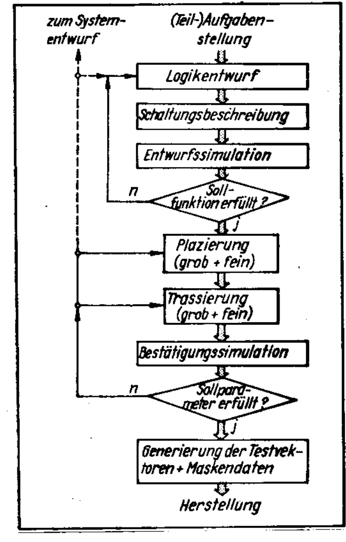


Bild 2 Entwurfsablauf bei Gate-Array-Schaltkreisen

Systemantwort in Form von Ausgangssignalen als Antwort auf seine Eingangsfolgen (-vektoren). Durch diese Dialogarbeitsweise sind notwendige Schaltungsänderungen und -anpassungen sofort eingebbar, womit eine iterative Angleichung der gegenwärtigen Istfunktion an die geforderte Sollfunktion realisierbar wird. Stimmen beide überein, wird die Entwurfssimulation durch den Entwerfer als „vollständig" definiert und beendet. Dabei werden in der Regel nur die zu realisierenden Hauptfunktionen geprüft und weiterhin Eingangsfolgen eingegeben, die zu keiner oder zu auszuschließenden Reaktionen führen sollen. Damit hängt von der Wahl der Eingangsvektoren die Aussagekraft der Simulationsergebnisse in entscheidendem Maße ab. Die zu nutzenden Eingangsvektoren sollten dabei sinnvollerweise parallel zum Schaltungsentwurf festgelegt werden, da in dieser Phase über die Funktion der Schaltung am intensivsten nachgedacht wird.

Diese Entwurfssimulation als eine zeitliche Logiksimulation wird oft – fälschlicherweise – statische Logiksimulation bezeichnet, weil die dynamischen Parameter der Schaltung noch nicht vollständig bekannt sind.
Nach der Entwurfsimulation folgt die Plazierung und die Trassierung. Diese werden im allgemeinen automatisch durchgeführt, wobei durch manuelle Eingriffe zielgerichtete Veränderungen und Vorgaben vorgenommen werden können. So ist zum Beispiel eine Vorplazierung oder die Festlegung von Nachbarfunktionselementen bei der für die Verarbeitungsgeschwindigkeit kritischen Teilschaltungen möglich. Um im Komplex Plazierung und Trassierung eine problemlose Verdrahtung der plazierten Funktionselemente zu ermöglichen, sollten nur etwa 80% der maximal möglichen Funktionselemente beim Entwurf genutzt werden. Nach erfolgreicher Plazierung und Trassierung, das heißt, nachdem die Verdrahtungsebenen (Alu- und Poly-Silizium-Leitbahnen) gänzlich entworfen sind, ist die genaue Länge und die Art aller Verbindungen fixiert. Damit können die durch Verdrahtungswiderstände und -kapazitäten entstehenden Verzögerungszeiten exakt – im Rahmen der Genauigkeit der benutzten Leitungsmodelle – berechnet werden.

Anschließend erfolgt die **Bestätigungssimulation**. Bei dieser werden auf der Basis der in der Bibliothek der Funktionselemente enthaltenen Verzögerungszeiten der Funktionselemente und der vorher berechneten Verzögerungszeiten der einzelnen Verbindungsleitungen die Verzögerungszeiten für alle Schaltungspfade berechnet. Bei dieser Bestätigungssimulation – oft auch als dynamische Logiksimulation bezeichnet – wird bei Worstcase-Bedingungen die maximal mögliche Taktfrequenz ermittelt.
Der Entwurfsvorgang wird abgeschlossen durch die Berechnung von Testvektoren für eine Strukturprüfung, die beim Hersteller erfolgt. Da der Hersteller über die **Funktion** des Gate-Array-Schaltkreises keine oder keine vollständigen Informationen besitzt, prüft er nur das richtige Funktionieren der Funktionselemente und deren Verschaltung; er führt eine **Struktur**prüfung durch, um gegenüber dem Anwender (= Entwerfer) den Nachweis zu erbingen, daß „sein Silizium" funktioniert. Dies ist in Werbeschriften durch den Satz beschrieben: Unser Silizium realisiert exakt Ihren Logikplan. Dies bedeutet, bei „richtigem", oder anders formuliert, die Verantwortung für die Funktion des Schaltkreises liegt beim Entwerfer! Dieser kann und muß durch die Simulation den Funktionsnachweis erbringen. Ist dies der Fall und ist die Strukturprüfung vollständig, kann eine Funktionsprüfung – wie sie bisher praktiziert wurde – bei Gate-Array-Schaltkreisen entfallen. Der Gate-Array-Schaltkreis erbringt mindestens die während der Simulation ermittelten Werte. Eine Funktionsprüfung ist nur dann sinnvoll, wenn zum Beispiel der Schaltkreis im Ausnahmefall mit einer höheren als der bei der Simulation ermittelten Frequenz betrieben werden soll.

Der Entwurf endet mit der Ausgabe obiger Strukturprüfungsdaten und der Daten des Verdrahtungslayouts auf periphere Speicher. Mit den Layoutdaten werden die Masken der Verdrahtungsebenen erzeugt und mit diesen die uniformen Transistorstrukturen auf dem Masteruntergrund individualisiert, wodurch die die individuellen Anwenderforderungen erfüllenden Gate-Array-Schaltkreise entstehen.

Abschließend soll nochmals formuliert werden, daß die Hauptaktivität bei einem Gate-Array-Schaltkreis-Entwurf *nicht* die Entwicklung von Schaltkreisen, sondern die Entwicklung und Verifikation der logischen Schaltung ist. Der Elektronikingenieur, der einen Gate-Array-Schaltkreis entwirft, muß dadurch keine detaillierten Kenntnisse über den Entwurf, sondern beispielsweise über die Makrobibliothek zur Realisierung *seiner* Schaltung besitzen.
Dies ist nur möglich, wenn ein leistungsfähiges CAD-System die umfangreichen Routinearbeiten durchführt. Ein derartiges CAD-System ist das Entwurfssystem **ARCHIMEDES** für das Gate-Array-System U 5200/5300 des VEB Forschungszentrum Mikroelektronik Dresden (ZMD). Dieses leistungsfähige System ist auf 32-Bit-Rechentechnik (z. B. K 1840) lauffähig.
Damit sehr frühzeitig und in großer Breite die Aus- und Weiterbildung zum Gate-Array-Entwurf praktiziert werden kann, wurde an der Technischen Universität Karl-Marx-Stadt das Entwurfssystem **PC-GAD** erstellt, das quelltextkompatibel zu ARCHIMEDES und auf 16-Bit-PCs lauffähig ist. Damit sind auch Teilentwürfe für kommerzielle Gate-Array-Schaltkreise am Arbeitsplatz des Entwerfers in der Industrie realisierbar.
Die einzelnen Komponenten dieses CAD-Systems und deren Nutzung werden in den nächsten Beiträgen vorgestellt.

Literatur

/1/ Reinert, D.: Entwurf und Diagnose komplexer digitaler Systeme. Berlin: Verlag Technik 1983

/2/ Elschner, H.; Möschwitzer, A.; Reibinger: Rechnergestützte Analyse in der Elektronik. Berlin: Verlag Technik 1977

/3/ Fügert, E.; Nehrkorn, P.: Dialogorientierte Netzwerkanalyse mikroelektronischer Schaltungen. Wissenschaftliche Schriftenreihe der TH Karl-Marx-Stadt (1984) 1

KONTAKT

Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022; Tel. 5613195

TERMINE

Wissenschaftliche Konferenz „Computerintegrierte Systeme für die industrielle Elektroenergietechnik"

WER? Technische Hochschule Leipzig, Sektion Elektroenergieanlagen
WANN? 26. bis 28. April 1989
WO? Leipzig
WAS?

- Elektrosicherheit und Zuverlässigkeit – Qualitätsbewertung
- Rechnerintegrierte Elektroantriebssysteme
- Konstruktion und Herstellung von Elektroenergieanlagen
- Entwurf und Betrieb von Netzen in Industrie und Kraftwerken
- CAD- und Prozeßleittechnik – Software für Elektroenergieanlagen

WIE? Interessenten wenden sich an Technische Hochschule Leipzig, Sektion Elektroenergieanlagen (WK '89), Karl-Liebknecht-Straße 132, Leipzig, 7030

Prof. Dr. Altmann

Fachtagung MIDOS '89

WER? AG(B) Inf./Dok. Potsdam und AG(B) Inf./Dok. Berlin der KDT
WANN? 30. Mai 1989
WO? Potsdam
WAS?
Anwendererfahrungen und Ausblick auf die Weiterentwicklung des Programmsystems MIDOS für die Nutzung in der wissenschaftlich-techischen Information und im Bibliothekswesen
WIE? Teilnehmerwünsche sind zu richten an: BVO Potsdam der KDT, SB Wissenschaft und Technik, Weinbergstraße 20a, Potsdam, 1560; Tel. 23426

Blackert

IX. Wissenschaftliche Konferenz für Energiewirtschaft „Forschung und Ausbildung für die Energiewirtschaft – 20 Jahre Technische Hochschule Zittau"

WER? Technische Hochschule Zittau
WANN? 7. bis 9. Juni 1989
WO? Zittau
WAS?

- Gestaltung von Energieversorgungssystemen
- Kohle- und Kernkraftwerke
- Hauptausrüstungen für Elektroenergie-Verteilungsanlagen
- Rationelle Energieausnutzung und Umweltschutz
- Instandhaltung und technische Diagnostik
- Automatisierung energetischer Prozesse
- Energie und Gesellschaft

WIE? Teilnahmemeldungen bitte an Technische Hochschule Zittau, Prof. Dr. G. Schumann, Theodor-Körner-Allee 16, Zittau, 8800; Tel. 610, Telex 284240

Vollack

EPROM-Programmiereinrichtungen aus dem IfAM Erfurt

Jürgen Wrenzitzki
Ingenieurbetrieb für die Anwendung der Mikroelektronik Erfurt

In der DDR werden Halbleiterbauelemente für verschiedene Mikroprozessorsysteme entwickelt und produziert. Die Bereitstellung und ständige Weiterentwicklung von LSI-Speicherbausteinen ist dabei unverzichtbarer Faktor. Seit dem Debüt des p-Kanal-EPROMs U 552 C ist in der DDR eine kontinuierliche Entwicklung auf dem EPROM-Sektor zu verzeichnen. Die Entwicklung ging über den 1-KByte-EPROM U 555 C (2708) in n-Kanal-Technologie bis hin zum U 2764 C, der den Besuchern der Leipziger Frühjahrsmesse 1988 präsentiert wurde. Mit der fortschreitenden Technologieentwicklung wird auch die künftige Entwicklung von Halbleiterspeichern einhergehen.

Die bisher durch den IfAM Erfurt vertriebene Programmleiterkarte EPROM 1/EPROM 1-Z fand einen breiten Anwenderkreis. In diesem Beitrag werden die neuen Programmiermodule EPROM 1-W und EPROM 2 vorgestellt, die seit dem IV. Quartal 1988 lieferbar sind.
Für die Programmierung von EPROMs, besonders der neuen Typen, wird seit 1987 durch den IfAM Erfurt ein EPROM-Programmiermodul für den PC 1715 vertrieben, der auch in K-1520-Rechnern einsetzbar ist /1/.
Die Entwicklung dieses Programmiermoduls EPROM 1/EPROM 1-Z im IfAM Erfurt hatte folgende Ziele:

- Schaffung einer universellen Hardwarelösung, die es ermöglicht, EPROMs der Typen 2708 bis 27256 zu programmieren
- Nutzbarmachung dieser Lösung speziell für den PC 1715 und darüber hinaus auch für K-1520-Rechner
- Unterstützung der Anwender durch ein breites Spektrum an mitgelieferter Steuersoftware für unterschiedliche Betriebssysteme und Rechner
- Keine Verwendung von Typensteckern oder dergleichen für die Typumschaltung.

Die geschaffene Lösung EPROM 1/EPROM 1-Z berücksichtigt alle Anwender, die den U 555 C und dessen Äquivalenztypen in ihrer Technik noch einsetzen. Rechner, die herstellerseitig bereits mit Programmiermoduln (für U 555 C) versehen waren (MRES, MC 80), können mit dem Programmiermodul des IfAM Erfurt ausgerüstet werden /2/.
Die bisher gesammelten Anwendererfahrungen weisen eine positive Resonanz aus, zeigen aber auch, daß besonders im Sinne der technischen Weiterentwicklung auf dem EPROM-Sektor die Einsatzfähigkeit der EPROM 1/EPROM 1-Z für neue Bauelemente begrenzt ist. Die technischen Erfordernisse, die Anwendererfahrungen mit dem Programmiermodul EPROM 1/EPROM 1-Z, die Verbreitung neuer Rechnertypen und damit verbunden neuer Betriebssysteme in der DDR waren Anlaß zu weiteren Arbeiten und zur Realisierung von zwei neuen Programmer-Versionen. Neben den auf internationaler Ebene industriell hergestellten EPROM-Programmiergeräten (die ausschließlich diesem Zweck dienen), die mit eigener „Intelligenz", Speicher zur Datenpufferung und Standard-Schnittstelle (V.24, RS 232 C ...) ausgerüstet sind, liefern einige Hersteller von Rechnerentwicklungssystemen systemeigene Programmer, die entweder direkt am Systembus oder über eine Schnittstelle in das System implementiert sind. Die Steuerung des Programmers und auch der notwendige Datenpuffer werden durch das System organisiert. Dadurch reduziert sich der erforderliche Hardwareaufwand im wesentlichen auf die Schnittstelle zum System und auf eine diskrete Schalter-/Treiberlogik zur Steuerung bzw. Umschaltung der Anschlüsse am Programmiersockel. Der Einsatz von Typensteckern (Characterizer) oder sogenannter Personality-Moduln zur Typumschaltung wird meist vermieden. Diese zuletzt genannten Faktoren und der Gedanke an eine flexible Lösung für verschiedene Rechner und Betriebssysteme führten im IfAM Erfurt zur Entwicklung von zwei ähnlichen EPROM-Programmer-Moduln, die sich durch ihre Schnittstellen unterscheiden.
Die EPROM 1-W nutzt (wie auch die EPROM 1/EPROM 1-Z) den Systembus des vorhandenen Rechners (K-1520-Rechner oder PC 1715) und wird über PIO-Bausteine U 855 D gesteuert. Die EPROM 2 ist für Rechner mit MS-DOS-kompatiblen Betriebssystemen vorgesehen und nutzt die (meist vorhandene) parallele Druckerschnittstelle (Centronics-Port). Der Datenaustausch zwischen Rechner und Programmer erfolgt seriell gepuffert. Beide Programmer EPROM 1-W und EPROM 2 erlauben die Programmierung von EPROMs der Typen 2716 bis 27256 in allen für sie gültigen Betriebsarten, sofern sie für den entsprechenden EPROM-Typ zugelassen bzw. vorgesehen sind. Die Software-Erweiterung auf die Typen 27512, 27513 bzw. für alle EPROMs im 28poligen Gehäuse, die der schon *historischen* Anschlußbelegung für 8-Bit-EPROMs entsprechen, ist ohne Hardwareänderungen möglich. Die Erweiterung des Typenspektrums auf EPROMs mit 16-Bit-Datenwortbreite ist konzeptionell möglich. Allerdings erfordert dies zusätzliche Hardware und wegen des notwendigen Einsatzes eines 40poligen Programmiersockels eine Abwendung vom Prinzip der *reinen Softwaresteuerbarkeit* des Programmers.
Unabhängig vom speziellen Programmertyp wird versucht, den Anwender in seinen Belangen hinsichtlich der Programmierung und Nutzung von EPROMs auf verschiedenen Rechnerbetriebssystemen durch die mitgelieferte Steuersoftware (plus Dokumentation) in hohem Maße zu unterstützen.
Es werden die Routinen

- *softwaremäßige Typauswahl* (Select, Exchange)
- *automatische Typauswahl* (Identify)

die EPROM-typischen Algorithmen

- *Leerkontrolle* (Blank Device Test)
- *Einlesen von Master-EPROM* (Fill)
- *Programmieren (in allen erforderlichen Modi)* (Program)
- *Vergleichen EPROM ↔ Puffer* (Compare)

und die für die File-Bearbeitung erforderlichen Software-Hilfsmittel

- *Laden von Disk(ette) (+ Verbinden von Files im Puffer)* (Load)
- *Schreiben auf Disk(ette) (+ Aufspalten von Files im Puffer)* (Save)
- *Ansehen des Pufferinhaltes bzw. der EPROM-Daten* (Display)
- *Verändern des Pufferinhaltes + Dateneingabe von Console* (Modify)
- *Beschreiben des Puffers mit Konstanten* (Konstant)

zur Verfügung gestellt.

Die Programmiermoduln werden als unbestückte, industriell hergestellte, durchkontaktierte Zwei-Ebenen-Leiterkarten angeboten, wobei die EPROM 1-W im K-1520-Format geliefert wird. Die Konfigurierung dieser Karte für den PC 1715 erfolgt durch den Anwender. Für die Innbetriebnahme der Leiterplatten erhält der Anwender alle erforderlichen Dokumentationen und Programme auf Anwenderdiskette plus Stromlaufplan und Bestückungsunterlagen.

Literatur

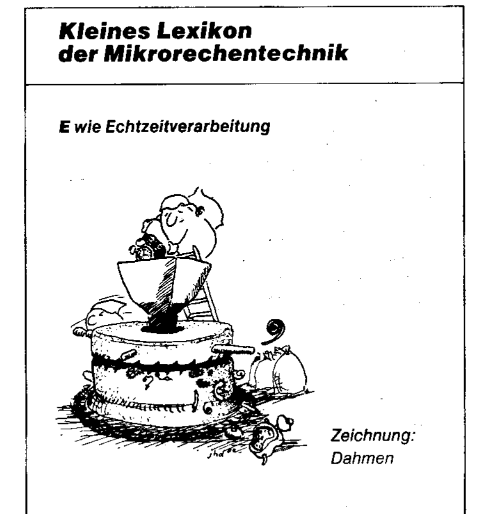
/1/ Programmiermodul für PC 1715. Radio, Ferns., Elektron. 36 (1987) 1, S. 4
/2/ MRES-Software für Programmiermodul EPROM 1. Radio, Ferns., Elektron. 36 (1987) 8, S. 475

KONTAKT

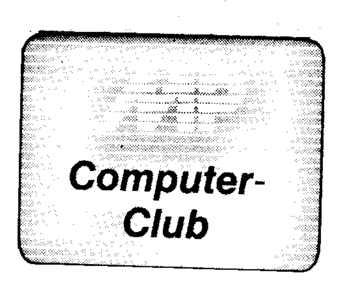
VEB Mikroelektronik „Karl Marx" Erfurt, IfAM Erfurt, Abt. CI 4 (Vertrieb) bzw. CI 2 (Technik), Rudolfstraße 47, Erfurt, 5010; Tel. 621 02/34 (CI 4) bzw. 621 02/23 (CI 2)

Kleines Lexikon der Mikrorechentechnik

E wie Echtzeitverarbeitung



Zeichnung: Dahmen



MSX – ein unbekannter Standard?

Michael Lennartz, Ludwigsfelde

Weitgehend unbekannt auf dem Markt der Heimcomputer in Westeuropa und den USA ist MSX, ein von führenden Firmen der Unterhaltungselektronik vereinbarter Standard für Computer des semiprofessionellen und des gehobenen Hobbybereichs. Konzept und Leistungsumfang dieser Geräte werden im folgenden Beitrag vorgestellt.

MSX steht für *Microsoft Super Extended Basic* und kennzeichnet einen Rechnerstandard, der Hardware und Betriebssystem umfaßt. Das 1982 von Microsoft ASCII Tokio entwickelte System, dem sich führende Firmen der Unterhaltungselektronik angeschlossen haben (u. a. Philips, Panasonic, Sony, Sanyo, Yamaha, Spectravideo und der Hersteller der LASER-Computer, Video Technology), basiert auf dem Mikroprozessor Z 80 A /1/. Die Weiterentwicklung MSX2 wurde 1984 vorgestellt, wobei zwischen MSX und MSX2 Aufwärtskompatibilität besteht. Hier wurde im wesentlichen die verbesserte Schaltkreisbasis berücksichtigt, indem ein neuer videotext-kompatibler Videoprozessor und ein größerer RAM-Bereich eingesetzt werden (vergleiche Tafel 1) /2/, /3/.

Das Besondere, das die Stärke von MSX ausmacht, ist der außerordentlich hohe Standardisierungsgrad. Nicht nur Mikroprozessor, Taktfrequenz und Videoprozessor sind genormt, sondern auch Tastatur und Steckplätze für die Erweiterungsmodule. Unabhängig vom Hersteller des Rechners, eines Zusatzmoduls oder eines Programms kann der Besitzer eines MSX-Rechners sicher sein, daß alle Komponenten paßfähig sind. Das ist anwenderfreundlich, hat aber auch positive wirtschaftliche Aspekte für die Herstellerfirmen. Die Konzerne der Unterhaltungselektronik, die nicht in erster Linie Computer herstellen, sondern eher um eine Symbiose zwischen Rechner und Audio- bzw. Videotechnik bemüht sind, müssen zu einem Rechner nicht noch eine Vielzahl von Zusatzmodulen und Software produzieren, sondern können sich auf ein einzelnes Produkt beschränken, das in die spezifische Erzeugnispalette paßt. Als Beispiel kann der Musikcomputer CX5M von Yamaha genannt werden, der einen eigenen Klangprozessor besitzt /4/.

Der auf ROM befindliche Basic-Interpreter ist eine Entwicklung von Microsoft und verfügt neben den von MBASIC bekannten Befehlen Anweisungen für Tonerzeugung und Grafik und unterstützt Multitasking-Anwendungen mit dem Befehl **ON INTERVAL GOSUB**. Durch die Verwendung von 14stelligen BCD-Zahlen werden Rundungsfehler vermieden. Obwohl als Heimcomputerstandard konzipiert und durch den Kassettenrecorder als primären Massenspeicher deutlich für den privaten Bereich ausgezeichnet, weisen die hohe Taktfrequenz und die Einführung des Diskettenbetriebssystems MSX-DOS darauf hin, daß MSX auch im professionellen Bereich einen Platz beanspruchen kann. Das Betriebssystem CP/M läßt sich nachladen, wodurch die umfangreiche CP/M-Software prinzipiell nutzbar wird, in der Praxis dürfte es bei MSX allerdings Schwierigkeiten durch das Bildschirmformat geben.

Tafel 1 Vergleich von MSX, MSX2 und BIC A 5105

	MSX	MSX2	A5105
Prozessor	Z 80 A	Z 80 A	UA 880 D
Taktfrequenz (typisch)	3,58 MHz	3,58 MHz	3,75 MHz
Videoprozessor	TMS9929A Texas Instruments	V9938 Yamaha	U 82720 VEB MME
Bildschirmformat	24 Zeilen zu ≥32 Zeichen	24 Zeilen zu 80 Zeichen	25 Zeilen zu 40/80 Zeichen
Video-RAM	≥ 16 KByte	≥ 64 KByte	64 K × 16 Bit
Grafikformat	256 × 192 Pixel	512 × 212 Pixel	640 × 200 oder 320 × 200 Pixel
Farben	16	256	16
RAM	≥8 KByte meist 64 KByte	≥64 KByte	64 KByte
Tongenerator	ja	ja	ja
Interfaces	Joystick Kassettenrecorder	Joystick Kassettenrecorder Maus Rollkugel Lichtgriffel	Joystick Kassettenrecorder Bus-Schnittstelle (im Grundgerät)
Basic	32 KByte ROM 144 Befehle	48 KByte ROM	48 KByte ROM RBASIC
Diskettenbetriebssystem	MSX-DOS (über Modul)	MSX-DOS (über Modul)	SCPX 5105

In den technischen Parametern ist der Bildungscomputer A 5105 mit den MSX-Rechnern vergleichbar. Der UA 880 D ist zum Z 80 A kompatibel, ebenfalls ist im A 5105 ein farbgrafikfähiger Videoprozessor mit 64 K × 16 Bit RAM für die Bildschirmsteuerung enthalten. Zur Grundausstattung gehören weiterhin ein 48-KByte-ROM-Basic (14 Stellen Rechengenauigkeit) und 64 KByte RAM sowie ein Diskettenlaufwerk und das Betriebssystem SCP.

Außerhalb von Japan ist der Erfolg von MSX bisher mäßig geblieben. Die Gründe dafür sind vielschichtig, aber sicher auch mit dem Erscheinen des IBM-PC und der Kompatiblen sowie dem folgenden Preisverfall für Geräte der 16-Bit-Klasse verbunden. MSX ist trotzdem ein Beispiel dafür, daß die Leistungsfähigkeit der 8-Bit-Technik zum Zeitpunkt des Übergangs zur 16-Bit-Technik noch nicht ausgeschöpft war.

Literatur

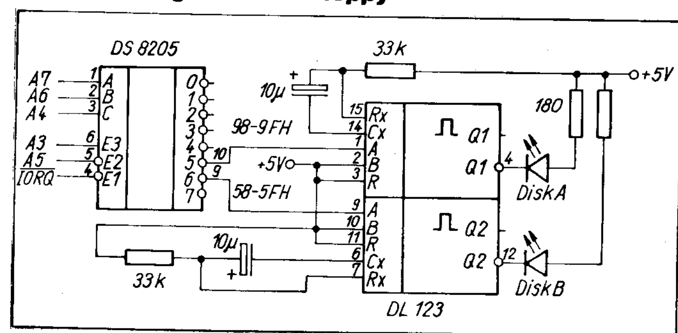
/1/ Hess, R.: Ganz schön flottes Innenleben. Chip, München (1985) 9, S. 238

/2/ Die zweite Generation. Chip, München (1986) 1, S. 184

/3/ Neues vom Standard. Chip, München (1985) 9, S. 72

/4/ Die aktuellen MSX-Computer. Chip, München (1985) 9, S. 74

Zugriffsanzeige für RAM-Floppy



Beim Betrieb von zwei RF-Karten nach MP 3/1988, Seite 74, war der nicht sichtbare Zugriff auf die RAM-Floppy durch den Rechner störend, da man nicht weiß, ob überhaupt noch etwas passiert, wenn der Rechner an einer etwas länger dauernden Aufgabe arbeitet.

Da an der RF-Karte keine Anzeige für einen Zugriff vorgesehen ist, würde, um eine Änderung der RF-Karte zu vermeiden, die Anzeige über den Systembus realisiert. Das heißt, daß die Zusatzschaltung auf einer kleinen Universalleiterkarte aufgebaut und auf einen freien Bussteckplatz gesteckt werden kann. Wird durch den Rechner auf eine der zu einer RAM-Floppy gehörenden Adressen zugegriffen, dekodiert der Dekoder ein Steuersignal für ein nachfolgendes Monoflop, welches die sehr kurzen Zugriffsimpulse auf ein „sichtbares" Maß verlängert.

Durch eine Änderung der Anschlüsse am Dekoder 8205 oder durch Belegung mit anderen Adreßleitungen können auch andere I/O-Grundadressen gewählt werden. Die in der Schaltung benutzten I/O-Grundadressen 98H und 58H resultieren aus dem Betrieb am Z 1013 und haben eine untergeordnete Bedeutung.

Rainer Brosig

Basic-Run im OS für die KC 87-Familie

Beim wiederholten Start des Basic-Interpreters im Betriebssystemmodus durch WBASIC bzw. REBASIC erfolgt die Abarbeitung der Initialisierungsroutine INITR mit einem anschließenden Sprung zur Routine EDIT des Interpreters. Die Analyse der Interpreterroutinen RUN und RUNMOD hat ergeben, daß das Basic-Kommando RUN im Betriebssystem realisiert werden kann, indem statt des Sprungs zur Routine EDIT nach INITR die Routine NEW2 gerufen und anschließend ein Sprung zu RUNMOD durchgeführt wird. Eine Implementierung für KC 85/1 und

```
OSRUN  Dok.

LINE         SOURCE

             PN     OSRUN
      ;
             ORG    7F00H
      ;
      INITR: EQU    0C669H
      RNMOD: EQU    0C854H                 ;=RUNMOD
      NEW2:  EQU    0C64FH
      ;
      ANF:   JMP    OSRUN
             DB     'RUN
             DB     0
      ;
      OSRUN: CALL   INITR                  ;Einstieg wie bei
                                           ;WBASIC bzw. REBASIC
             CALL   NEW2
             JMP    RNMOD
      ;
             END
```

KC87 kann, wie im angegebenen ASM-Listing aufgezeigt, erfolgen. Durch Eingabe von RUN ENTER kann damit ein Basic-Programm sowohl im OK als auch im Betriebssystemmodus gestartet werden. Diese Möglichkeit eignet sich für den Start einer Kombination von Basic- und Maschinencodeprogrammen, welche als Datei vom Typ COM erzeugt wurde und für die nach dem bzw. durch das Laden eine Initialisierung des Basic-Notizspeichers der Kombination entsprechend erfolgt. Für den Start des Basic-Teils der Kombination ist lediglich der Teil ab OSRUN im ASM-Listing in den Maschinencodeprogrammteil der Kombination aufzunehmen und die Startadresse beim Erzeugen der COM-Datei entsprechend zu wählen.

Literatur

/1/ Völz, H.: Subroutinen des BASIC-Interpreters von den KC-Rechnern. Mikroprozessortechnik 1 (1987) 6, S. 182

/2/ Völz, H.: Universelle Nutzung des BASIC-Interpreters. Mikroprozessortechnik 1 (1987) 7, S. 221

Dr. Lothar Boltze

REDABAS-Tip

Sinusfunktion

```
*--------------------------------------------------------------------
* SINUS.PRG / 11.09.87 / REALISIERUNG DER SINUS-FUNKTION
*--------------------------------------------------------------------
* DAS BERECHNUNGSVERFAHREN IST IN DER BEDIENUNGSANLEITUNG DES
* MINIREX 75,VEB ROEHRENWERK MUEHLHAUSEN ZU FINDEN.
* DER WERTEBEREICH IST ZWISCHEN 0 UND 90 GRAD !
* DIE GENAUIGKEIT IST BESSER ALS 0.01 %.
*--------------------------------------------------------------------
* EINGANG: ME1 = < WINKEL (NUMERISCH,IN GRAD >
*
* INTERN : MI1...MI3 = HILFSVARIABLEN
*
* AUSGANG: MA1 = < ERGEBNIS >, MA1 = SIN ( ME1 )
*--------------------------------------------------------------------
STORE ME1/90 TO MI1
STORE 1.5706268 * MI1 TO MA1
STORE 3 TO MI2
STORE 1 TO MI3
DO WHILE MI2 > 0
 STORE MI3 * MI1 TO MI3
 STORE MI2-1 TO MI2
ENDDO
STORE MA1+(-0.6432292 * MI3) TO MA1
STORE 2 TO MI2 •
DO WHILE MI2 > 0
 STORE MI3 * MI1 TO MI3
 STORE MI2-1 TO MI2
ENDDO
STORE MA1+0.0727102 * MI3 TO MA1
RELEASE MI1,MI2,MI3,ME1
RETURN
*--------------------------------------------------------------------
```

Zur Berechnung der Sinusfunktion (Bild) wird folgende Gleichung benutzt:

$\sin w = a x + b x^3 + c x^5$

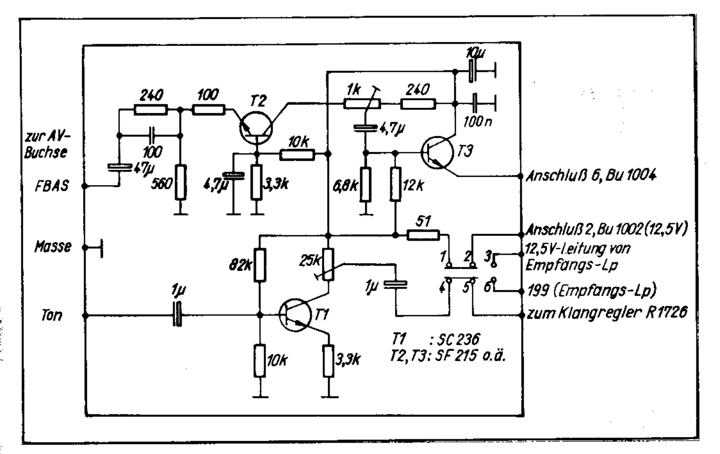
$$\text{mit } x = \frac{w \text{ (in Grad)}}{90°} \quad \text{und} \quad a = 1.5706268,\; b = -0.6432292,\; c = 0.0727102$$

Thomas Steffens

FBAS-Anschluß für Farbfernsehgeräte der Serie 3000

Der Anschluß eines Computers an einen Fernsehempfänger über das FBAS-Signal bringt gegenüber dem HF-Eingang einen merklichen Qualitätsgewinn. Farbfernsehgeräte der Serie 3000 (Colortron, Colorett) des VEB Fernsehgerätewerke Staßfurt sind nur für den Anschluß audiovisueller Geräte über Antenneneingang vorbereitet. Die im folgenden beschriebene Lösung ist ohne Änderung an den Leiterplatten und ohne Ausbau von Baugruppen des Empfängers nachrüstbar.

Die vom Computer kommenden Ton- und FBAS-Signale werden durch die Transistorstufen T1 und T2 nachverstärkt. Mittels der Einstellregler lassen sich die Pegel so anpassen, daß beim Umschalten der Betriebsarten keine Nachregelung der Lautstärke und des Kontrastes nötig ist. Die Einkopplung des FBAS-Signals erfolgt über T3 am Emitterwiderstand der Video-Vorverstärkerstufe T2392, der am Anschluß 6 des Steckverbinders Bu1004 liegt. Die Umschaltung des Videokanals wird über die 12,5-V-Betriebsspannung vorgenommen. Die 12,5-V-Zuführung für den HF- und ZF-Teil des Empfängers (Empfangsleiterplatte) wird am Anschluß 2 des Steckverbinders Bu1002 abgetrennt und an den Umschaltkontakt 3 geführt. Umschaltkontakt 2 wird mit dem Anschluß 2 der Bu1002 verbunden. Da die 12,5-V-Betriebsspannung unabhängig von der Betriebsart am Kontrast- und Helligkeitsregler benötigt wird, ist eine zusätzliche Verbindung zwischen Anschluß 2 der Bu1002 und der oberen Anschlußfahne des Helligkeitsreglers R1363 (schwarze Leitung) herzustellen. Die 12,5-V-Verbindung zwischen Helligkeits- und Kontrastregler (schwarz) bleibt bestehen, aber die ursprüngliche 12,5-V-Zuführung (schwarz) zum Kontrastregler ist zwischen diesem und dem Kabelbaum durchzutrennen.



Im Tonkanal werden die Signalwege am Eingang des Klangreglers umgeschaltet. Dazu wird das geschirmte Kabel am Anschluß 199 auf der Empfangsleiterplatte abgelötet und zum Umschaltkontakt 5 verlängert. Zwischen Leiterplattenanschluß 199 und Umschaltkontakt 6 ist eine geschirmte Verbindung herzustellen.

Die zusätzliche Schaltung läßt sich leicht an der Plasteplatte unterbringen, die die Antennenbuchse des Empfängers trägt. Unterhalb der Antennenbuchse befinden sich zwei vorbereitete Ausbrüche für den Einbau von Anschlußbuchsen. Eine dieser Öffnungen kann für die Unterbringung des Umschalters genutzt werden. Nach unten ist noch genügend Raum für eine kleine Leiterplatte mit der Verstärkerschaltung vorhanden.

Literatur

/1/ Schornack, W.; Sonka, U.: Videorecorderanschluß für Fernsehempfänger. Radio, Ferns., Elektron., Berlin 35 (1986) 7, S. 439

/2/ Blochwitz, E.: Bild- und Tonanschluß mit RGB-Qualität für Farbfernsehgeräte der Serie 4000. Mikroprozessortechnik, Berlin 2 (1988) 2, S. 61

/3/ Stromlaufplan Colortron, Colorett Serie 3000 (1980). VEB Fernsehgerätewerke Staßfurt

Günter Salzmann

Kopieren und Retten von Bildschirminhalten

Die vorgestellte Routine für den KC85/3 ermöglicht die Erzeugung speicherplatzgünstiger Kopien von Bildschirminhalten mittlerer Dichte. Dabei werden je Bild nur etwa 2 bis 5 KByte benötigt. Erst bei sehr dichten Bildern steigt der RAM-Bedarf über die Originalgröße des Pixel-RAM (10 KByte). Die erzeugten Kopien sind unabhängig vom ursprünglichen COPY und auf jeder Adresse lauffähig.

Arbeit mit COPYR

Das Programm liegt im vom Basic-Interpreter (ROM-Basic) nicht genutzten Bereich, lediglich der Platz für die Kopie muß reserviert werden.

Aufruf aus Basic:
DOKE 0, aaaa : REM Adresse der Kopie
POKE 2, n : REM Bildnummer
CALL * 200 : REM Ruf COPYR

Aufruf aus CAOS-Menü:
% COPY aaaa n

Aufruf aus COM-Routinen:
LD HL, aaaa
LD DE, n
CALL 0021EH

Nach Erzeugung der Kopie wird aus HL ihre Endadresse als Hex-Zahl ausgegeben.

Retten der Kopie:
% SAVE aaaa eeee

Aufruf der Kopie:
BASIC: CALL * aaaa
MENU: RECOPYn
COM: CALL aaaa + 016 H

Altes und neues Bild werden ODER-verknüpft.

Heiko Schlittermann

200	CD	18	F0	FD	E5	2A	00	00	03E1
208	ED	5B	02	00	CD	1E	02	FD	0334
210	E1	CD	1B	F0	C9	00	00	7F	0401
218	7F	43	4F	50	59	01	D5	E5	0375
220	11	73	02	EB	01	36	00	ED	0295
228	B0	FD	E1	ED	53	15	02	D1	04B6
230	3E	30	83	FD	77	14	21	FF	0399
238	7F	06	00	23	04	E5	11	FF	02A1
240	A7	A7	ED	52	E1	28	1D	3E	03F1
248	FF	B8	7E	C2	5C	02	E5	2A	0464
250	15	02	70	23	77	23	22	15	017B
258	02	AF	47	E1	FE	00	C2	4E	03E7
260	02	C3	3B	02	2A	15	02	36	0179
268	00	23	CD	03	F0	1A	CD	03	02CD
270	F0	2C	C9	CD	18	F0	CD	0F	0496
278	F0	0E	00	CD	1B	F0	C9	7F	041E
280	7F	52	45	43	4F	50	59	20	0271
288	01	CD	0F	F0	18	00	11	1B	0211
290	00	19	EB	21	FF	7F	06	00	02A9
298	1A	FE	00	C8	13	4F	09	1A	0265
2A0	13	4E	B1	77	18	F0	E1	E5	0457
2A8	C9	00							00C9

Indizierte Variablen und Felder unter REDABAS

Dr. Thomas Streubel
Präsidium der Kammer der Technik

Bei Anwendung des Datenbanksystems REDABAS können in der 8-Bit-Version bekanntlich bis zu 64, in der 16-Bit-Version 256 Variablen genutzt werden. Da ihre zügige sequentielle Selektion relativ umständlich zu erreichen ist, wurden bereits in /1/ und /2/ entsprechende schnelle Zugriffsprogrammvarianten erarbeitet.

Der triviale Fall einer Arbeit mit indizierten Variablen ist in Anlehnung an /1/ im Bild 1a dargestellt. Es werden beispielhaft 54 Variablen (VARI1 bis VARI54) generiert und auf den Wert Null gesetzt. Dabei ist deutlich die Rolle des Makrooperators erkennbar, mit dem die Indexziffern als Zeichenkette an den Variablennamen angehängt werden. Zur Lösung des dabei entstehenden Problems, die wechselnde Stellenzahl dieses Wertes ohne ein führendes, fehlererzeugendes Leerzeichen zu erreichen, wurde ein entsprechender Test in der **IF**-Schleife eingesetzt. Durch eine weitere Umgestaltung dieses Größentests im IF-Zweig ist sofort eine Erweiterung auf über 100 indizierte Variable für die 16-Bit-Version realisierbar. Die dann entstehende geschachtelte **IF**-Anweisung stellt wie ihr gezeigtes kleineres 8-Bit-Pendant (auch im Falle der Nutzung der **CASE**-Klausel) eine relativ große Folge von Einzelanweisungen dar, die durch den REDABAS-Interpreter recht laufzeitintensiv abzuarbeiten ist.

Akzeptiert man dagegen die Variablen auch in der Form VARI01, VARI02 bis VARI62 (8-Bit-Version) bzw. VARI001, VARI002 bis VARI254 (16-Bit-Version), so schaffen jeweils die einfachen einzeiligen Anweisungen

store $ (str((&ind + 101), 3, 0), 2, 2) to ind (8-Bit)

bzw.

store $ (str((&ind + 1001), 4, 0), 2, 3) to ind (16-Bit)

den gleichen Effekt! Im Bild 1b ist die sich durch Nutzung der entsprechenden $-Anweisung dann stark reduzierende Grundschleife für die 8-Bit-REDABAS-Version gezeigt. Durch diese Substring-Anweisung wird der Laufindex künstlich um den Wert 101 erhöht, um ihn dadurch exakt längendefiniert und inkrementiert auf die Index-Zeichenkette *IND* abzubilden. Implementiert man in einer weiteren Schleife eine entsprechend indizierte **RELEASE**-Anweisung, so werden die erzeugten Variablen schnell gelöscht. Leider erfolgte in /1/ bezüglich der erwähnten Selektion von indizierten Datenfeldern keine Beispieldarlegung. Im Bild 2 wird an Hand der Berechnung eines Prozentwertes und dessen Ablage gezeigt, wie die in (allen) Datensätzen vorhandenen Felder der KRIT01 bis KRIT23 einer Quelldatei DATEI1 einem sequentiellen Zugriff unterworfen werden. Dabei erfolgte das Einschreiben der berechneten (Prozent-)Werte in die (ebenfalls für die Indizierung entsprechend strukturierten) Felder DATFELD01 bis DATFELD23 einer gleichgroßen Zieldatei. Der Makrooperator dient hier nicht nur der Indizierung, sondern deutet die hohe Flexibilität der Nutzung eines solchen Moduls an. Das dritte Bild stellt eine diesbezüglich interessante Variation der Mehrfachanwendung von Grundschleifen dar. Es werden in den 2 Beispielzyklen durch Anwendung des Makrooperators sowohl die Berechnungsalgorithmen (hier einfache Prozent- und Promilleberechnungen), die Quelldateien (DATEIQA und DATEIQB), die Zieldateien (DATEIZX und DATEIZY) als auch die darin befindlichen Ergebnisablagefelder (KENNFELD und BEIFELD) der aktuellen Zieldatei variiert. Dabei wird außerdem in den *kleinen* Schleifen über eine weitere Variable (DS) gesichert, daß (alle) 33 bzw. 47 Datensätze der entsprechend gewünschten Berechnung unterzogen werden. Es ist denkbar, mit 2 weiteren Variablen nach dem Öffnen von Quell- bzw. Zieldatei unterschiedlichste Datensätze anzuwählen, um auf diese Art und Weise gezielt selektiv und/oder mit einem gewissen Datensatzoffset zu arbeiten. Mit einem solchen Modul ist es möglich, maximale Flexibilität zur Realisierung unterschiedlichster Algorithmen zu erreichen.

Der interessierte Leser wird mit den hier dargestellten Beispielen der unüblichen Indizierung auch in 8- und 16-Bit-REDABAS-Datenbanksystemen in die Lage gebracht, dieses wirksame Hilfsmittel in eigenen Programmodulen rationell einzusetzen. Hervorzuheben ist eine dann damit fast gleichzeitig erreichbare außerordentlich hohe Flexibilität der Nutzung einmal erstellter Grundmodule, die im nicht unwesentlichen Maße zur Qualitätserhöhung und zur Mehrfachnutzung von Software beitragen können.

Literatur

/1/ Matzke, B.: Indizierte Variablen unter REDABAS. Mikroprozessortechnik, Berlin 2 (1988) 4, S. 104
/2/ Hilbert, A.: Diagrammdarstellung auf dem PC 1715. Mikroprozessortechnik, Berlin 2 (1988) 5, S. 152

✉ KONTAKT ☏

Kammer der Technik, Präsidium, Clara-Zetkin-Straße 115/117, Berlin, 1086; Tel. 2265200

Bild 2

```
store 23 to kritanz
store 'datei1' to qelldatei
store 'datei2' to zieldatei

select primary
use &qelldatei
select secondary
use &zieldatei
select secondary
set linkage on
do while .not. eof
   store '1' to ind
   goto &ind
   do while &ind <= kritanz
      replace datfeld&ind with (p.krit&ind/p.bezwert)*100
      store $(str((&ind+101),3,0),2,2) to ind
   enddo
   skip
enddo
select primary
use
select secondary
use
release quelldatei,zieldatei,kritanz,ind
```

```
* Beispiel der Arbeit mit indizierten Variablen VARI1 bis VARI54

*********  Beginn einer indizierten Grundschleife ****
           store '1' to ind
           store val(ind) to i
           do while i <= 54
*********     Anweisungsteil                      ********
              store 0 to vari&ind
*********     Ende des kurzen Anweisungsteils ********
              store val(ind) to i
              store i+1 to i
              if i < 10
                 store str(i,1,0) to ind
              else
                 store str(i,2,0) to ind
              endif
           enddo
           release i,ind
*********  Ende der indizierten Grundschleife ********
```

Bild 1a

Bild 1b

```
* Beispiel der Arbeit mit indizierten Variablen VARI01 bis VARI54
* unter Nutzung einer wesentlich reduzierten Grundschleife

*********  Beginn einer indizierten Grundschleife ****
           store '1' to ind
           do while &ind <= 54
*********     Anweisungsteil                     ********
              store 0 to vari&ind
*********     Ende des kurzen Anweisungsteils ********
              store $(str((&ind+101),3,0),2,2) to ind
           enddo
           release ind
*********  Ende der indizierten Grundschleife ********
```

Bild 3

```
store 1 to anzahl
do while anzahl <= 2
   if anzahl=1
      store '((bsberges/bsbestges)*100)' to krit
      store 'dateiqa' to quelldatei
      store 'dateizx' to zieldatei
      store 'kennfeld' to zielfeld
      store 33 to ds
   endif
   if anzahl=2
      store '(((sksi+mksi)/(sksz+mksz))*1000)' to krit
      store 'dateiqb' to quelldatei
      store 'dateizy' to zieldatei
      store 'beifeld1' to zielfeld
      store 47 to ds
   endif
   use &quelldatei
       store '1' to iind
       do while &iind <= ds
          store &krit to f&iind
          store $(str((&iind+101),3,0),2,2) to iind
          skip
       enddo
   use
   use &zieldatei
       store '1' to iind
       do while &iind <= ds
          replace &zielfeld with f&iind
          store $(str((&iind+101),3,0),2,2) to iind
          skip
       enddo
   use
   store anzahl+1 to anzahl
enddo
```

PC-Hauptspeichererweiterungen

Im Jahre 1981 präsentierte IBM den ersten *Personal Computer* mit dem Betriebssystem PC-DOS. Dieser Computer verfügte damals noch über 16 KByte Hauptspeicher. Auch sein Nachfolger, der PC/XT, hatte 1983 erst 64 KByte. Damals erschien die theoretische Grenze des PC-DOS (die kompatiblen Rechner benutzen MS-DOS) von 640 KByte bei der damals verfügbaren Software, der Kapazität von Speicherschaltkreisen und ihren Preisen als eine Grenze, die in absehbarer Zeit nicht zu erreichen ist. Mit der schnellen Verbreitung des IBM-PCs und seiner Kompatiblen stiegen jedoch sehr schnell die Anforderungen an die Software und an die Benutzerschnittstellen. Die damit in Verbindung stehende rasche Erhöhung der Produktion von Speicherschaltkreisen senkte deren Preis erheblich. Damit wurde ein Kreislauf in Gang gesetzt, der innerhalb von wenigen Jahren dazu führte, daß die ferne 640-KByte-Grenze immer näher rückte und zu einem echten Hemmnis wurde. Welche Möglichkeiten daraufhin geschaffen wurden, diese Grenze zu überspringen, wollen wir im folgenden kurz erläutern.

Der unter DOS (gemeint ist PC- bzw. MS-DOS) vom Anwender nutzbare Speicher von 640 KByte wird als **Conventional Memory** (konventioneller Speicher) bezeichnet. Der Prozessor 8086/88 kann mit seinen 24 Adreßleitungen (physisch) nur 1 MByte adressieren, wovon die oberen 384 KByte durch DOS reserviert sind. Der 80286 und der 80386 können physisch 16 MByte bzw. 4 GByte adressieren.

Im Unterschied zum 8086/88 verfügen der 80286 und der 80386 über die Möglichkeit der virtuellen Speicheradressierung. Dadurch können 4 GByte bzw. 64 TByte adressiert werden. Hierfür besitzen diese Prozessoren eine Speicherverwaltungseinheit auf dem Prozessorchip. Diese **Memory Management Unit** (MMU) erfüllt zwei Aufgaben (siehe Bild 1). Sie rechnet erstens die virtuellen (logischen) Adressen des Prozessors in die physischen Adressen um (das Programmieren mit logischen Adressen ist hardwareunabhängig). Zweitens reagiert die MMU, wenn der physische (wirklich vorhandene) Speicher voll ist. Sie lagert dann den Hauptspeicherinhalt auf einen Massenspeicher (meist Festplatte) um und holt ihn bei Bedarf in den Hauptspeicher zurück.

Da immer mehr Anwendungen an die durch den 8086/88 begründete 640-KByte-DOS-Grenze stießen, entwickelten die Firmen Lotus, Intel und Microsoft (LIM) im Jahre 1985 den **Expanded Memory Standard** (EMS), der in der Version 3.2 die Nutzung eines *8 MByte* großen *Datenspeichers* außerhalb des konventionell von DOS adressierbaren 1-MByte-Adreßraumes ermöglicht (siehe Bild 2). Die Daten werden über ein 64 KByte großes Fenster gelesen bzw. geschrieben.

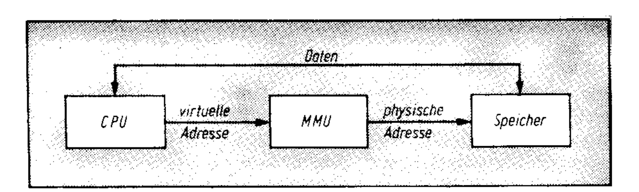


Bild 1 Umwandlung der virtuellen in die physische Adresse durch die MMU

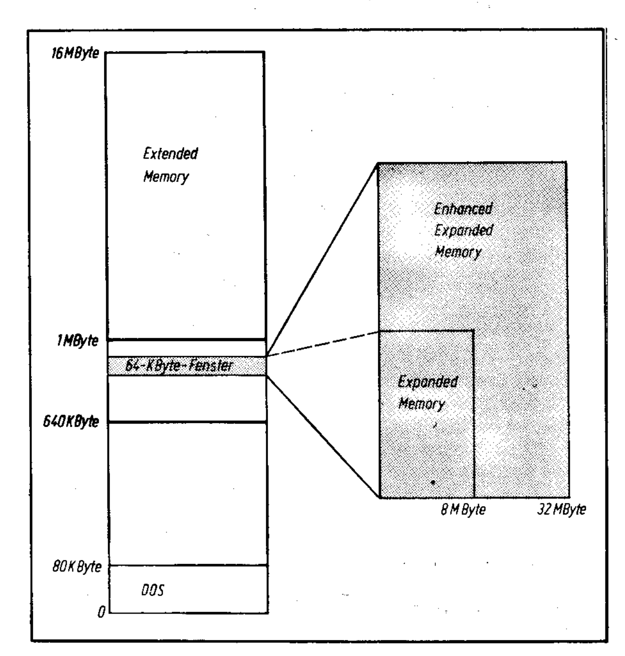


Bild 2 Konventioneller Speicher und Speichererweiterungen

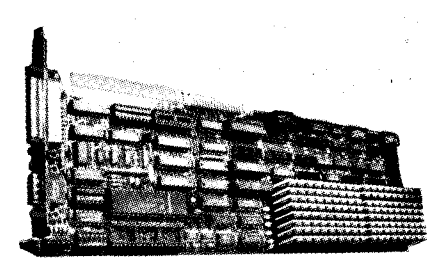


Bild 3 Erweiterungskarte Elite 16 mit maximal 16 MByte RAM

Um diesen Adreßraum auf *32 MByte* zu erhöhen, begründeten Ashton-Tate, Quadram und AST Research den **Enhanced Expanded Memory Standard** (EEMS). Aber nicht nur *Daten*, sondern auch *Programme* werden über das 64 KByte große Fenster gelesen bzw. geschrieben. Durch die Auslagerung von Programmen in den 32-MByte-Adreßraum wurden die Voraussetzungen für Multitasking (quasi gleichzeitiges Abarbeiten mehrerer Aufgaben) geschaffen.

Die Software, die den Zugriff über dieses Fenster auf den Expanded Memory bzw. Enhanced Expanded Memory erlaubt, wird als **Expanded Memory Manager** (EMM) bezeichnet.

Lotus, Intel und Microsoft schlossen sich dem EEMS an und nannten ihr neues Produkt EMS Version 4.0 (bzw. LIM 4.0). Es beinhaltet alle Leistungsmerkmale von EMS 3.2 und EEMS und ist zudem noch zu EMS 3.2 kompatibel. Unter DOS 4.0 wird EMS 4.0 genutzt, um Treiberprogramme aus dem konventionellen Speicher auszulagern.

Die Prozessoren 80286 und 80386 sind durch ihre Möglichkeit der physischen Adressierbarkeit von mehr als einem Megabyte an sich nicht an die 640-KByte-Grenze von DOS gebunden. Deshalb ist der Zugriff über ein 64-KByte-Fenster (also nur 64-KByte-weise) für diese Prozessoren zu umständlich. Für sie wurde nun der **Extended Memory Standard** (XMS) geschaffen. Dadurch kann der gesamte 16-MByte-Adreßraum des 80286 direkt (ohne Umweg über ein Fenster) nutzbar gemacht werden (siehe Bild 2). PC-Anwender können diesen XMS jedoch effektiv erst unter dem neuen Betriebssystem OS/2 einsetzen (unter DOS ist der Zugriff auf den Extended Memory wesentlich langsamer). Die XMS-Version 2.0 von Lotus, Intel, Microsoft und AST Research (Oktober 1988) verfügt über einen Treiber, der nicht nur kommerzieller Software, sondern auch Anwenderprogrammen den Zugriff auf den Extended Memory gestattet.

Verschiedene Hersteller bieten nun Erweiterungskarten für PC-Steckplätze an. Diese können meistens wahlweise nach EMS 3.2, 4.0 oder XMS arbeiten. Der Speicher kann bei einigen Karten auch in einen Teil für Expanded Memory und einen Teil für Extended Memory aufgeteilt werden (das ist sinnvoll, wenn beispielsweise im Multitasking unter OS/2 eine Task DOS ist). Üblicherweise werden Speichererweiterungskarten mit 2, 8 oder 16 MByte RAM angeboten.

Besitzt der Rechner, der mit einer solchen Karte bestückt wird, auf der Hauptplatine noch nicht die vollen 640 KByte RAM oder ist der RAM auf der Hauptplatine wesentlich langsamer als der RAM der Erweiterungskarte, dann kann die Erweiterungskarte den Speicher bis 640 KByte auffüllen (*backfilling*) oder ihn völlig ersetzen.

Hauptvertreter der Hersteller von Erweiterungskarten ist die Firma Intel. Ihr *Above Board* aus dem Jahre 1985 hat inzwischen schon drei Nachfolger: *Above Board Plus* und *Above Board Plus I/O* (für den PC/AT) sowie *Above Board/2 Plus* (für das PS/2). Alle drei Karten haben eine Speicherkapazität von 2 MByte bei Verwendung von 256-KBit-Chips und von 8 MByte bei Verwendung von 1-MBit-Chips; sie ermöglichen EMS 4.0 oder XMS. Auf dem Above Board Plus I/O befindet sich zusätzlich ein paralleler Druckerport und bis zu zwei seriellen Schnittstellen.

Das Board *Elite 16* von Profi Systems Inc. (siehe Bild 3) kann in PCs der AT-Klasse verwendet werden und erreicht mit 256-KBit-Schaltkreisen 4 MByte und mit 1-MBit-RAMs 16 MByte (auch eine kombinierte Bestückung ist möglich). Als Extras können in diesem Speicher ein Disk-Cache, ein Druckerspooler und/oder eine RAM-Disk enthalten sein. Die Aufteilung des Speichers in Expanded Memory und Extended Memory ist möglich.

Eine andere interessante Lösung für XT/AT-kompatible Rechner stellt die *RYBS-AMS-HIcard* von Frisbie Data Systems dar. Sie erhöht die unter DOS nutzbare Speicherkapazität zwar nur auf 896 KByte, verschiebt aber hardwaremäßig die magische 640-KByte-Grenze um 64 KByte auf 704 KByte (und benötigt nur einen kurzen PC-Erweiterungssteckplatz). Der Speicher zwischen der 704-KByte- und der 896-KByte-Grenze kann für eine RAM-Disk, einen Druckerspooler und/oder speicherresidente Programme eingesetzt werden (siehe auch MP 8/1988, S. 255).

MP-Hk

Drucker, Farbtücher, Reinigungsdiskette

Folgende Neuerervorschläge wurden am Karl-Weierstraß-Institut für Mathematik der AdW der DDR realisiert und stehen zur Nachnutzung zur Verfügung:

NV 1/88 Druckeranschluß PD 27000 (Videoton) an PC-XT und Kompatible
NV 2/88 Anpassung der Farbtuchkerne und -halterungen für Farbtücher des Typs PD 27060 an PD27090 (Videoton)
NV 3/88 Reinigungsdiskette für 5¼"-Laufwerke
NV 4/88 Centronics-Schnittstellenumschalter für Mehrfachnutzung von Druckern

Karl-Weierstraß-Institut für Mathematik, BfN, Mohrenstraße 39, Berlin, 1086; Tel. 2 07 73 01

Dr. Auert

TURBO-DATA

TURBO-DATA ist ein einfach anzuwendendes System von Prozeduren, die mittels Includefiles in Turbo-Pascal-Programme eingebunden werden und die direkt (d. h. ohne temporäre Dateitransformationen) auf REDABAS-Dateien zugreifen.
Es ergeben sich alle Vorteile einer höheren Programmiersprache, wie mehrfach schnellere Programmlaufzeiten und erweiterte Programmiermöglichkeiten.
Die Prozeduren erfüllen im wesentlichen folgende Funktionen:
- Datensätze lesen, schreiben, löschen
- Schlüssel suchen, weiterstellen, einfügen, löschen u. a.

IFA Automobilwerke Ludwigsfelde, Abt. TVF, Ludwigsfelde, 1720; Tel. 6 20 86

Golze

Textverarbeitung und Kassettensteuerung für KC 87

Für den Kleincomputer KC 85/1 bzw. KC 87 wird ein Textverarbeitungssystem angeboten, welches die Eingabe, Korrektur, den Druck und die Archivierung beliebiger Texte ermöglicht. Das als Fullscreeneditor implementierte System erlaubt mittels komfortabler Bildschirmarbeit die Eingabe von Text über Tastatur und Kassette, das Streichen und Einfügen von Zeilen, Zeichen und ganzen Textabschnitten und die Erzeugung der Rechtsbündigkeit des Textes. Es stehen Funktionen zum Verschieben und Kopieren ganzer Textabschnitte zur Verfügung. Da der Bildschirm des KCs nur 40 Spalten/Zeile darstellen kann, wird bei Erreichen des Zeilenendes der gesamte Bildinhalt nach links geblättert, so daß auch Tabellen komfortabel erarbeitet werden können. Die Bedienung des Textverarbeitungssystems erfolgt über die Computertastatur oder die Tastatur einer angeschlossenen Schreibmaschine. Das Textverarbeitungssystem und der Schreibmaschinenanschluß sind im Technikum LAURA nachnutzbar.
Weiterhin wurde ein steuerbares Kassettengerät an den KC 85/1 angeschlossen. Dieser Anschluß erlaubt eine inhaltsverzeichnisorientierte Dateiarbeit auf der Kassette, wie sie sonst nur bei angeschlossenen Diskettenlaufwerken üblich ist. Das Inhaltsverzeichnis wird am Anfang der Kassette abgespeichert, so daß alle Programme die von ihnen benötigten Dateien selbständig auf dem Band suchen und laden bzw. diese an freien Stellen auf dem Band abspeichern. Die Hard- und Software zur Ansteuerung eines Kassettengerätes vom Typ SK 3000 kann im Technikum nachgenutzt werden. Weiterhin gehören zum Nachnutzungsumfang komfortable Kassettenkomprimier- und Kopierprogramme.

Friedrich-Schiller-Universität Jena, Sektion Technologie für den WGB, Technikum LAURA, Ernst-Thälmann-Ring 32, Jena, 6900; Tel. 8 22 21 32

Reinhardt/Slowik/Entreß

Kopplung ESER – A 7100

In unserer Einrichtung, dem VEB IFA Getriebewerke Brandenburg, ist es gelungen, eine direkte Kopplung zwischen der Lochbandeinheit EC 7903 (M) eines ESER 1035 und einem Arbeitsplatzcomputer A 7100 zu realisieren.
Der Hardwareaufwand ist für den A 7100 sehr gering. Es wurde nur die Centronics-Schnittstelle modifiziert und über ein 15adriges Kabel mit einem Paneel an der LBE verbunden.
Dieses Paneel nimmt 9 kleine Leiterkarten auf, welche die LBE an die veränderten Übertragungsbedingungen zum A 7100 anpassen. Die LBE ist auch weiterhin für die Lochbandarbeit nutzbar. Die direkte Übertragung wird durch eigene Programme gesteuert. Es wird eine praktische Datenübertragungsgeschwindigkeit von 1000 bis 2000 Zeichen pro Sekunde erreicht. Die Datensicherheit wird gewährleistet. Die bisherige praktische Nutzung hat den Nachweis erbracht, daß die Kopplung einfach und sicher zu nutzen ist. Zur Anpassung der unterschiedlichen Datei- und Speicherungsarten wird von uns das System DDS. des VEB Datenverarbeitung Schwerin genutzt.

VEB IFA Getriebewerke Brandenburg, Abt. LO, Wilhelm-Bahms-Str. 9, Brandenburg, 1800; Tel. 57 36 01

Py/Be

Grafischer Editor CAD88

Mit CAD88 können symbolorientierte Zeichnungen einfach erstellt, geändert, gespeichert und auf externe Ausgabegeräte (Plotter, Drucker) ausgegeben werden. CAD88 ist lauffähig auf dem A 7100 unter SCP 1700 und auf dem A 7150 und DCP 1700. Als Eingabegeräte können entweder das grafische Tablett K 6405 oder die alphanumerische Tastatur eingesetzt werden. Zur Ausgabe der Bilder werden grafikfähige Drucker (K 6313, K 6314 u. ä.) und Plotter (K 6411, K 6418, SPL) unterstützt.
Der grafische Editor dient der Erstellung und einer anwenderfreundlichen Änderung von Schalt- und Übersichtsplänen, Netzplänen, Programmablaufplänen u. a. und orientiert vor allem auf Zeichnungen mit vorrangig symbolorientiertem Charakter. Im Hauptmenü werden die Grundfunktionen *Bild laden, Bild bearbeiten, Bild sichern, Bild ausgeben, Bildschirm löschen* und *Symboldatei pflegen* angeboten. Unter *Bild bearbeiten* stehen dem Nutzer die grafischen Funktionen Linie (Linienart, Linienbreite), Kreis, Kreisbogen, Kreissegment, Fläche (Polygonzug füllen, Schraffurarten) und Texte (Texthöhe in mm, Textwinkel) zur Verfügung.
Das Editierprogramm bietet die Möglichkeit der Vereinbarung von Symbolen (z. B. Schaltzeichen, Schriftfelder, Rahmen). Symbole werden durch einen Bezugspunkt definiert, mit dessen Hilfe das Symbol positioniert, gedreht, vergrößert und verkleinert werden kann. Zur Erhöhung der Genauigkeit der Arbeit können Ausschnitte gewählt und vergrößert werden. Ebenfalls sind im Editierprogramm das Merken und Verschieben des Ausschnittfensters enthalten.
Mit der Funktion *Korrektur* wird ein anwenderfreundlicher Änderungsdienst angeboten. Durch Anzeige von Markern, zugehörig zu einzelnen grafischen Elementen, Texten bzw. Symbolen, kann jedes grafische Gebilde direkt identifiziert und mit ihm operiert werden. Als Korrekturfunktionen stehen das Löschen, Verschieben und Kopieren zur Verfügung. Für diese Operationen können mehrere Elemente gleichzeitig ausgewählt werden.
Während der Bildbearbeitung ist das Hinterlegen eines Linienrasters möglich. Der Abstand der Linien wird in Millimetern abgefragt, wobei dieses Maßverhältnis der gleichen Größenordnung auf externen Ausgabegeräten bei einem Ausgabeverhältnis von 1:1 (Standard) entspricht. Die Originalgröße der Bilddarstellung beträgt max. 400 × 500 mm. Die Angabe eines Maßstabes während der Ausgabe mit *1:N, N:1* oder einer Kombination bietet weitreichende Möglichkeiten für die Wahl des Größenverhältnisses der gewünschten Zeichnung.

Institut für Energieversorgung, Abt. SE, Zeunerstr. 83a, Dresden, 8027; Tel. 4 65 04 24

Dr. Pötschke/Kießling

Druckerprogramm für A 7100

Der auf dem AC A 7100 implementierte Textprozessor WS unterstützt nur wenige der üblicherweise benötigten Druckerfunktionen. Aus diesem Grunde wurde ein Programm geschaffen, mit dem Texte gedruckt werden können, die unter WS editiert wurden. Die folgenden Kommandos zur Druckersteuerung werden realisiert:

1. Fettdruck (dreimal übereinander)
2. Breitdruck
3. Unterstreichen (auch bei Leerzeichen)
4. Durchstreichen
5. Indizieren
6. Exponieren
7. komprimierte Normalschrift
8. normale Schreibdichte
9. Umschalten Zeichengenerator:] } | \ { [~
10. Rückschalten Zeichengenerator: Ü ü ö Ö ä Ä ß
11. Schrägdruck
12. Überdrucken durch ein nachfolgendes Zeichen: a ≠ b
13. Farbbandumschaltung
14. Druckpause

Das DOT-Kommando. pl wurde neu belegt. Die übrigen DOT-Kommandos sind fast vollzählig implementiert. Darüber hinaus besteht die Möglichkeit, Grafikdateien zu lesen und zwischen den Text einzuschieben. Voraussetzung ist ein grafikfähiger Drucker und die grafische Schicht SCP-GX, die zuvor geladen worden sein muß.
Die Grafiken sind mit GEDIT/M16 aufzubauen und abzulegen. Das Programm steuert den Wechsel zwischen alphanumerischer und grafischer Ausgabe, der beliebig oft stattfinden kann.

Technische Universität Karl-Marx-Stadt, Sektion Vorkurse, Ernst-Schneller-Straße 138, Breitenbrunn, 94 34

Dr. Winkler

FastSearch

Übliche Datenbanksysteme zur Offline-Literaturrecherche basieren auf dBASE oder REDABAS. Neben dem Vorteil einer vergleichsweise einfachen Programmierung sind damit gravierende Nachteile verbunden:
– Die unterschiedliche Länge der erfaßten Zitate einerseits und die geforderte feste Satzstruktur andererseits bedingen Kompromisse. Es geht entweder Speicherplatz verloren oder Information, in Extremfällen sogar beides gleichzeitig.
– Die Recherche im Datenbestand erfolgt nicht allzu schnell.
– Die Eingabe der Recherchekriterien erfolgt „blind", das heißt, man weiß zu Beginn der Recherche nicht, ob die Suche z. B. nach einem bestimmten Autor überhaupt Zweck hat.
– Ohne Festplatte müssen bereits mittlere Datenbestände auf mehrere Disketten verteilt werden, wodurch eine Suche sehr erschwert wird.
Das an unserer Sektion entwickelte Programm FastSearch arbeitet mit drei Wörterbüchern für Autoren, Titel- und Schlagworte sowie ergänzende Angaben, z. B. die Zeitschriftentitel. Die erfaßten Literaturzitate selbst bestehen lediglich aus Verweisen auf die Wörterbücher. Die Wörterbuchverwaltung erfolgt selbstverständlich automatisch während der normalen Programmnutzung. Dieses Prinzip weist momentan noch den Nachteil der völligen Inkompatibilität der erfaßten Daten mit gängigen kommerziellen Datenbanksystemen auf, jedoch auch eine Reihe wesentlicher Vorteile:
– Die Neuerfassung von Zitaten wird stark vereinfacht. Anhand der Wörterbücher versucht das Programm ständig zu „erraten", was der Bediener gerade eingeben möchte. Dadurch

genügen z. B. zur Eingabe eines Autorennamens durchschnittlich zwei Buchstaben.
- Die Recherche erfolgt mit 100 bis 500 Zitaten/Sekunde.
- Die Eingabe der Recherchekriterien ist sehr elegant möglich. Besitzer einer Maus können fast ausschließlich damit operieren.
- Die Größe der erfaßten Zitate ist nahezu unbegrenzt. Momentan sind willkürlich 30 Autoren, 200 Titel- und 50 Schlagworte festgelegt worden.
- Auf einer 360-KByte-Diskette können 2000 bis 3000, auf einer 720-KByte-Diskette 5000 bis 7000 Zitate erfaßt werden.
Das Programm setzt einen Rechner mit dem Betriebssystem MS-DOS, 512 KByte RAM und Grafikmöglichkeit (mindestens IBM CGA) voraus. A7150 und EC 1834 erfüllen diese Forderungen.
FastSearch besteht aus 5 Dateien mit derzeit 160 KByte Gesamtgröße.
Interessenten erhalten kostenlos eine Demoversion sowie eine Demonstrationsdatenbank. Voraussetzung ist eine transportsicher verpackte Leerdiskette.

Karl-Marx-Universität, Sektion Chemie, Talstraße 35, Leipzig, 7010; Tel. 71 65 3 95

Dr. Haeßner

Programmgenerator HWSGEN 1

Ziel der Entwicklung war die Reduzierung bzw. vollständige Abschaffung der aufwendigen Arbeit mit Bildschirmmasken und deren Umsetzung in ein Programm.
Die Abarbeitung des Programms erfolgt in 2 Abschnitten:
• Aufbau der Bildschirmmaske und Parametertabelle (Full-Screen-Modus, Sicherheitsabzug der Maske und Parametertabelle für spätere Weiterbearbeitung, Speichern der Maske als Textfile für Dokumentationen)
• Generierung der Quelltextfiles xxxxxxxx.PAS und .INC
.INC: in sich abgeschlossene Prozedur zur Bearbeitung der Maskeneingabe (Bildschirmsteuerung, geprüfte Eingabe, Korrekturen während und nach der Eingabe eines Bildes)
.PAS: alle Anweisungen darüber hinaus, sowie Muster für eventuell zusätzlich erforderliche Prüfroutinen.
Diese Quellfiles können sofort mit Turbo-Pascal übersetzt werden. Das fertige Erfassungsprogramm der jeweiligen Bildschirmmaske legt die Daten im File xxxxxxxx.DAT auf Diskette ab. Für alle in Turbo-Pascal zugelassenen Datentypen werden während der Eingabe nur die jeweils zulässigen Zeichen akzeptiert. Die universelle Eingabeprozedur *unieingabe* wird nur nach Eingabe eines gültigen Wertes verlassen. Kommentarausgabe und Editieren werden durch die Prozedur realisiert.
Wurde in der Maske die Eingabe eines Datums erkannt, wird zusätzlich die Prozedur *unidat* eingekettet, welche alle Prüfungen für ein gültiges Datum enthält.
Anwendung:
- Erstellung kompletter Erfassungsprogramme
- Erstellung der Eingabeprozedur für Masken oder Teilmasken zur weiteren Verwendung im eigenen Programm.
Bei Anforderung der Dokumentation ist eine Diskette erforderlich.

VEB Hebezeugwerk Suhl, Abt. LO, Am Sehmar 1, Suhl 6000; Tel. 2 30 13 App. 30

Dunkel

Stücklistenprozessor auf KC 85/3

Für die Kleinrationalisierung in Konstruktionsbüros bieten wir ein Basic-Programm auf KC 85/3 (32 KByte Arbeits-RAM) zum Aufstellen, Bearbeiten und Drucken von Stücklisten an.
Im Arbeitsspeicher des Rechners können gleichzeitig beliebig viele Stücklisten mit beliebiger Hierarchietiefe aufgebaut werden, wenn die Gesamtzahl aller Positionen (Datensätze) 99 oder 255 nicht übersteigt.
Integriert wurde ein Formelinterpreter zur Unterstützung von Masseberechnungen und die Ermittlung der Gesamtmasse über alle Positionen einer Stückliste. Die Datenstruktur der Stücklisten und die Druckparameter sind leicht an unterschiedliche Belegsätze anpaßbar.
Die durch die Hardware bedingten zeitkritischen Aktionen beanspruchen:
- Laden des Programms: ca. 1:30 min
- Einlesen/Abspeichern eines Datenblockes mit 100 Positionen: ca. 1:45 min.
Alle Editierfunktionen dagegen werden mit vertretbarer Geschwindigkeit ausgeführt.

VEB Entwicklung und Rationalisierung der ÖVW des Bezirkes Magdeburg, Wilhelm-Raabe-Straße 8, Magdeburg, 3080; Tel. 3 57 65

Putzke

Datenerfassungs- und Statistikprogramm

Das Programm **EDISTAT** dient zur Erfassung (Editieren) und der Bearbeitung (statistische Analysen) von numerischen Daten. Es ist auf IBM-kompatiblen Rechnertypen lauffähig. Um es ebenfalls den Nutzern der Betriebssysteme CP/M 80 sowie CP/M 86 zugänglich zu machen, sind etwas abgerüstete Versionen für diese Betriebssysteme erhältlich. Der Einsatz des Grafikmoduls für die CP/M 80-Version wurde nicht vorgesehen. Auf Grund der Gesamtgröße des Programms von zirka 200 KByte (ohne Grafikmodul zirka 150 KByte) machte sich die Nutzung der Overlay-Technik erforderlich, um genügend Speicher für die Daten zur Verfügung zu haben. Damit konnte der Speicherplatzbedarf des Programms zur Laufzeit auf zirka 50 KByte (ohne Grafik zirka 20 KByte) reduziert werden.
EDISTAT ermöglicht die Arbeit im deutsch- und englischsprachigen Dialog, wobei die Sprache auch während des Programmlaufes wahlweise geändert werden kann.
Mit EDISTAT werden folgende statistische Berechnungen und Analysen realisiert:
- Mittelwerte, Varianz und Standardabweichung
- Aufstellen der Häufigkeitstabelle
- einfache Varianzanalyse
- Diskriminanzanalyse (in Vorbereitung)
- Berechnung von Testquantilen verschiedener Verteilungen
- lineare Regression
- Matrix der Kovarianzen auf Korrelationskoeffizienten
- Kolmogorow-Smirnow-Test auf verschiedene Verteilungen, wobei sich noch weitere Analysen in Vorbereitung befinden. Für einige dieser Berechnungen ist unter den oben genannten Bedingungen (Betriebssystem) eine grafische Auswertung möglich. Die Ergebnisausgabe erfolgt wahlweise auf Bildschirm und/oder Drucker.

Betriebssystemabhängig bietet das Programm eine Reihe von Serviceleistungen, wobei die nachstehend aufgeführten Funktionen in allen Versionen verfügbar sind:
- Lesen und Schreiben der Daten von/auf Diskette
- Anzeige des Inhaltsverzeichnisses von Disketten
- Transformieren von Daten, Berechnung neuer Werte mittels eines eingegebenen Algorithmus
- Erzeugen arithmetischer Reihen, um laufende Eingaben zu vereinfachen
- Erzeugen von Zufallszahlen verschiedener Verteilungen.

Ingenieurhochschule für Seefahrt, R.-Wagner-Str., PF 100, Rostock, 2530; Tel. 5 72 94

Fischer

Hardware-RESET für Schneider-PC und Software für Rechnerkommunikation

Die PC 1512/1640 verfügen nicht über die Möglichkeit, ein **Hardware-RESET** auszulösen. Bei der Programmerprobung und besonders bei der Fehlersuche in der Hardware des Rechners wirkt sich dieser Mangel störend aus. Wird bei einer Fehlfunktion die Tastatur des Rechners inaktiv, bleibt nur noch das Ausschalten des Gerätes.

Mit Hilfe einer kleinen Zusatzplatine wird bei Bedarf ein systemgerechtes **RESET**-Signal erzeugt.

Das Softwarepaket **DUESE** gewährleistet den Dateitransfer zwichen Rechnern, die mit den seriellen Schnittstellen IFSS und V.24 ausgerüstet sind. Bedingung ist, daß diese Rechner hardwareseitig auf den Schaltkreisen SIO und CTC basieren. Die Datenübertragungsgeschwindigkeit beträgt bei V.24 9600 Baud, bei IFSS 4800 Baud. Routinen zur Fehlerbehandlung und Kontrolle der Datenübertragung sind implementiert.

Die Programme sind in Turbo-Pascal geschrieben und werden bisher auf den Rechnern K8915, A7100 und A7150 (SCP, DCP) verwendet. Unter DCP wird die hierarchische Dateistruktur unterstützt. Im Gegensatz zu Kermit oder TLC erfolgt die Bedienerführung dialogorientiert. Die Anwendung von DUESE ist bei der Konvertierung SCP-DCP oder bei unterschiedlichen Diskettenformaten vorteilhaft.

Ostthüringer Molkereikombinat, Industriestraße 1, Gera, 6500; Tel. 5 12 88

Schumann

SD 1154 an IBM-kompatiblen Computern

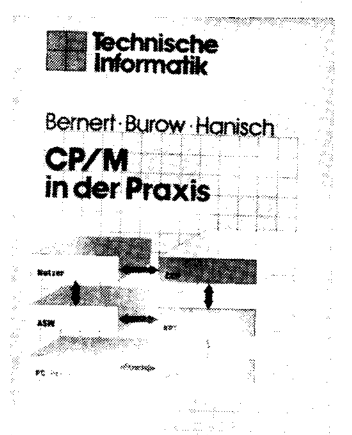
Um die an vielen Einrichtungen noch vorhandenen robusten Drucker vom Typ SD 1154 (und solche mit dem gleichen Interface) auch mit 16-Bit-PCs nutzen zu können, wurde an der Sektion Physik der KMU Leipzig eine aufwandsarme Kopplung dieser Geräte entwickelt. Voraussetzung ist ein IBM PC/XT/AT-kompatibler Computer mit einer vollwertigen Centronics-Schnittstelle. Folgende *Leitungen* werden benötigt: Datenleitungen, Statusleitungen (Eingabe) und Steuerleitungen (Ausgabe). Die Status- und Steuerleitungen werden ausgenutzt, um verschiedene Leitungen des SD 1154 zu bedienen (CR, LF, FF usw.). Die normalen Datenleitungen steuern /DATA 1 bis /DATA6 sowie RD (Rotdruck) am SD 1154. Daraus ergibt sich der notwendige Aufwand beim Nachnutzer: Es ist eine Verbindung zwischen einem Centronics-Stecker und einem Stecker für den SD 1154 zu schaffen. Nach dem Starten des zugehörigen *Treibers* verhält sich der SD 1154 auf der Ebene von Betriebssystem und Anwenderprogrammen wie ein ganz normaler Drucker. Er kann dazu über eine beliebige der drei möglichen Centronics-Schnittstellen (LPT1, LPT2, LPT3) angeschlossen und angesprochen werden, insbesondere können gleichzeitig ein hochwertiger Matrixdrucker und der SD 1154 im System aktiv sein.
Im *Installationsprogramm* wird der Inhalt einer Zeichenübersetzungstabelle abgefragt, um insbesondere IBM-Sonderzeichen auf dem SD 1154 erkennbar darstellen zu können. Der Original-Zeichengenerator des SD 1154 besitzt einen sehr beschränkten Zeichensatz, meist nur lateinische und kyrillische Großbuchstaben. Insbesondere für die hier beschriebene Anwendung lohnt sich daher die Umrüstung auf Kleinbuchstaben. Es wird die Nachnutzung der Neuererleistung *Umrüstung des Zeichengenerators des SD 1154 auf Kleinbuchstaben empfohlen.* Es handelt sich um eine kleine Leiterplatte für einen EPROM, die anstelle des alten Zeichengenerators eingesetzt wird. Die Nachnutzung der Neuererleistung zur Kopplung eines SD 1154 an einen IBM-kompatiblen Computer umfaßt den Inhalt einer nachnutzereigenen Diskette, sowie einige Informationsblätter.
Die Herstellung des Verbindungskabels obliegt dem Nachnutzer; in Ausnahmefällen können unabhängig von der Nachnutzung (bei Lieferung entsprechender Stecker) einzelne Verbindungskabel angefertigt werden.
Die Nachnutzungsgebühr für den Anschluß des SD 1154 an einen IBM-kompatiblen Computer beträgt etwa 500,– Mark. Zur Beschleunigung des Verfahrens empfiehlt es sich, eine formatierte Diskette (IBM-Format 360 KByte, sonst genau angeben!) und eine verbindliche Bestellung zu schicken.

KMU Leipzig, Sektion Physik, WB Halbleiterphysik, Elektroniklabor, Linnéstraße 5, Leipzig, 7010; Tel. 6 85 84 06

Gürtler

CP/M in der Praxis

von J. Bernert, M. Burow und C. Hanisch, Berlin: VEB Verlag Technik 1988, 224 S., 126 Bilder, 59 Tafeln, Broschur, DDR 23,– M, Bestell-Nr. 553 961 6



CP/M ist mit seinen Kompatiblen in der DDR wie weltweit das am weitesten verbreitete Betriebssystem für 8-Bit-Rechner. Spät bereichert das nun vorliegende Buch die spärlich vorhandene Literatur in der DDR über dieses Betriebssystem. Es ist kein Lehrbuch, bietet aber eine Beschreibung wichtiger Kommandos und Dienstprogramme, die auch für den Anfänger verständlich ist. Der Umfang der Einzelheiten übersteigt sicher das, was ein Anfänger braucht, und bietet deshalb auch dem geübten CP/M-Nutzer noch Neues.
Im einzelnen werden die residenten Kommandos des CP/M, die transienten Kommandos STAT, PIP, D, SYSGEN und die Dienstprogramme POWER für Dateiarbeit und WordStar für Textverarbeitung erläutert, alles Kommandos und Programme, die fast jeder CP/M-Nutzer braucht. Dabei werden die Syntax der Komponenten wie die Wirkungen, mögliche Fehlersituationen und Behebung der Fehler behandelt. Tafeln fassen das Wesentliche zur Arbeit mit jedem einzelnen Kommando zusammen. Verwendete Begriffe werden definiert, allerdings nicht immer vor der Benutzung. Viele praktische Tips helfen, Klippen der erklärten Komponenten und des Rechners zu umgehen.
Das Sachwörterverzeichnis erleichtert die Suche nach speziellen Kommandos bzw. Unterkommandos im Text.

Renée Mundstock

Der UNIX-Werkzeugkasten – Programmieren mit UNIX

von Brian W. Kernighan und Rob Pike, Carl Hanser Verlag, München Wien, 1987, 402 Seiten, ISBN 3-446-14273-8

Der Hanser Verlag in der BRD hat sich in den vergangenen Jahren mehrfach bei der Herausgabe wichtiger deutschsprachiger UNIX-Standardwerke verdient gemacht. Einen „Klassiker" auf diesem Gebiet stellt zweifellos die Übersetzung des Buches „The UNIX Programming Environment" von Kernighan und Pike (Original: Bell Inc. 1984) dar.
In der Fachbuchreihe „PC-professionell" erschienen, stellt dieses, von zwei mitverantwortlichen Schöpfern des Betriebssystemkonzeptes UNIX verfaßte, Buch auf etwa 400 straff genutzten Seiten eine sehr gute Einführung in das Betriebssystem UNIX und in eine Reihe seiner Standardwerkzeuge dar.
In acht gut strukturierten und dank der ausgezeichneten deutschsprachigen Übersetzung auch flüssig lesbaren Kapiteln wird folgender Stoffumfang behandelt:

- elementare Systembenutzung (Anmelden, Post, Kommandosyntax)
- UNIX-Dateiverwaltungssystem (Dateien, Directories, Zugriffsschutz, Inodes, Dateihierarchie)
- Shell-Kommandosprache (Struktur, Metazeichen, Argumente und Parameter, Kontrollstrukturen, Redirektion)
- Filer (grep, sort, sed, awk)
- Shell-Programmierung (cal, which, while, until ...)
- Standard-Bibliotheksbenutzung (stdio)
- Systemaufrufe des UNIX-Betriebssystems
- Programmentwicklungswerkzeuge (lex, yacc, make)
- Textaufbereitung (nroff, troff, ms, tbl, eqn).

Besonders gut hat dem Rezensenten gefallen, daß jedes Kapitel mit einem Abschnitt „Geschichte und Literaturhinweise" endet. Hier findet der Leser wertvolle Querverweise und weitergehende Literaturstellen zu den behandelten Themen.
Fazit der Rezension: „Der UNIX-Werkzeugkasten" von Kernighan/Pike ist ein ausgezeichnet geschriebenes Einführungswerk in das UNIX-Betriebssystemkonzept, das besonders durch die hohe Kompetenz seiner Autoren besticht.

Dr. L. Claßen

Maschinensprache des IBM-PC/AT

und Kompatibler in der Praxis. 2., erweiterte Auflage. Von Isa Brors, Heidelberg: Hüthig Verlag 1988, 338 S., ISBN 3-7785-1630-2



Assemblerprogrammierung ist traditionsgemäß im Bereich der system- und hardwarenahen Programmierung angesiedelt. Moderne PC-Implementierungen von Hochsprachen gestatten das direkte Einbinden von Maschinencode in Form von INLINE-Anweisungen. Durch diese Möglichkeit herausgefordert, gerät manch Anwendungsprogrammierer in die Versuchung, den Schritt zur hardwarenahen Programmierung und damit auch zum Assembler zu erproben. Dieses Buch kann ihm dabei ein guter Begleiter sein, denn es wartet mit einer hohen Informationsdichte und einer diesem Leserkreis angemessenen Erläuterung grundlegender Fachtermini der Assemblerprogrammierung auf. Eine Reihe praxisrelevanter und interessanter Programmbeispiele hebt es deutlich vom Handbuchcharakter ab. Nichtsdestotrotz ist das Buch aber auch als Nachschlagewerk für den Fortgeschrittenen interessant. Daß der Verfasser recht gründlich zur Sache geht, zeigt sich auf den ersten 20 Seiten. Hier wird zunächst etwas zu Dualzahlen, BCD-, Oktal- und Hexadezimaldarstellung von Zahlen ausgeführt. Daran schließt sich eine Beschreibung der inneren Struktur des 8088 an. Hier werden der Registersatz, die Adreßerzeugung und die Adressierungsarten dargestellt. Zur Erklärung der Adressierungsarten bedient sich der Autor kleiner Grafiken, die die möglichen Varianten auch für Anfänger transparent machen.
Der Befehlssatz des 8088 wird in Handbuchmanier und unterteilt in herkömmlicher Art und Weise in Arithmeitk-, Logikbefehle usw. vermittelt. Dieser Teil ist auch durch die gestalterische Form als Nachschlagewerk ausgelegt. Der Assembler selbst wird recht kurz, aber dem praxisorientierten Anliegen des Buches angemessen, beschrieben. Hier findet man den prinzipiellen Aufbau und die Pseudo- und Makro-Pseudo-Operationen erläutert. Es schließt sich ein Gliederungspunkt an, der mit „Das erste Programm ..." überschrieben ist. Unter diesem Punkt wird nun der Prozeß von der Erstellung des Quelltextes mit einem Editor über die Assemblierung und das Linken bis zum lauffähigen Programm etwas näher – jedoch recht formell – betrachtet. Es folgen auf etwa 70 Seiten sorgfältig ausgewählte Beispiele, die den besonderen Wert des Buches ausmachen, und für die – soweit es sich anbot – auch der hardwareseitige Hintergrund kurz umrissen wird. Das Spektrum der Beispiele erstreckt sich dabei von der Ausgabe einer Zeichenkette auf Bildschirm über Tastaturabfrage, Datum-und-Zeit-Routinen, Druckerausgabe, Arithmetik, Listenverarbeitung bis hin zur Programmierung des internen Lautsprechers und der Hercules-Grafikkarte. In vertiefender Form werden zur Grafik last but not least eine recht ansprechende Linienprozedur und eine Prozedur zum Zeichnen von Kreisen vorgestellt.
Nach einem kurzen Eingehen auf die Merkmale von COM- und EXE-Files wendet sich der Autor dem Herz des PC/AT zu – dem 80286. Dieser etwa 50 Seiten umfassende Abschnitt unterteilt sich in die Beschreibung der Spezifik des Real-Address- und des Protected-Mode des AT-Prozessors. Die von diesem Prozessor zusätzlich bereitgestellten Befehle werden in der gehabten Manier nüchtern und kurz beschrieben. Darüber hinaus wird anschaulich die Arbeitsweise des 80286 im Protected-Mode dargestellt. Der Autor bedient sich dazu zahlreicher Grafiken. Den Abschluß des Buches bildet ein kurzes Eingehen auf Hardware-Erweiterungen in Form der Darstellung der Hardwarecharakteristik einer parallelen und einer seriellen Schnittstelle.
In einem 16 Seiten umfassenden Anhang A wird eine tabellarische Übersicht zu den Interrupts eines PC gegeben. Eine sicher notwendige und nie ausreichende Ergänzung bei Büchern mit vorliegender Thematik. Auch kommt hier ganz deutlich die Tatsache zum Ausdruck, daß Maschinensprache stets in enger Verbindung mit einem Betriebssystem – wie in diesem Falle des DOS – zu sehen ist. Hier wären ausführlichere Hinweise zur Orientierung des Lesers hilfreich. Ein weiterer Anhang enthält eine Zusammenstellung der ASCII-Zeichencodes, der Tastatur-Codes und eine Information über Bildschirm-

Tagungsbände CAQ '88

Die Wissenschaftlich-Technische Gesellschaft für Meß- und Automatisierungstechnik veranstaltete im März 1988 in Leipzig das Internationale Messesymposium *Rechnergestützte Qualitätssicherung CAQ '88*. Einen guten Einblick in die wachsende Vielfalt von CAQ geben die Tagungsbände
Plenarvorträge (331 Seiten)
Poster (185 Seiten)
KDT-INQUAMESS-Software-Paket (252 Seiten).
Die Tagungsmaterialien fanden auf dem Symposium hohe Anerkennung aufgrund ihrer hervorragenden inhaltlichen und drucktechnischen Gestaltung. Von Fachleuten wurde eingeschätzt, daß mit diesen Tagungsbänden das gegenwärtig modernste Schriftgut zur rechnergestützten Qualitätssicherung in der DDR vorgelegt wurde. Die große Nachfrage und die Einschätzung im In- und Ausland waren für uns ausschlaggebend, eine zweite Auflage vorzubereiten. Für einen Unkostenbeitrag von 80,– M können die o. g. Materialien bestellt werden über
Kammer der Technik, Präsidum – WGMA, PSF 1315, Berlin, 1086.

Müller

Attribute, die sich leider auf den Schwarz-/Weiß- beziehungsweise Monochrommonitor beschränkt. Was man am Ende des Buches dann vergeblich sucht, ist eine Zusammenstellung weiterführender und vertiefender Literatur.
Das Buch ist kurzum ein Tip für die Einsteiger in die Assemblerprogrammierung, die mit den Grundlagen der Computerei prinzipiell vertraut sind. Die dargebrachten Beispiele wecken das praktische Interesse und regen dazu an, sich auch vertiefende Kenntnisse über die betriebssystemspezifische und stark hardwareorientierte Programmierung anzueignen. *Th. B.*

Serielle Busse: neue Technologien, Standards, Einsatzgebiete

von Conrads u. a. (Hrsg.), vde-Verlag 1987, 286 Seiten

Die Autoren führen den Leser in ein aktuelles Material ein. Äußerst praxisnahe Erfahrungen des Autorenkollektivs erleichtern neben der systematischen und oft detaillierten Darstellung die Einarbeitung in die Problematik.
Bedingt durch die außerordentlich stürmische Entwicklung auf dem Gebiet der seriellen digitalen Kommunikation, die ihre Grundlage in der zunehmenden Integration findet, hat sich eine nur schwer zu überblickende Menge von Standards, Normen und auch Systemen herausgebildet.
Eine Beurteilung der Systeme selbst, vor allem aber eine Beurteilung des Einsatzes derartiger Systeme, ist damit erschwert. Dieser Fülle des Stoffes nähern sich die Autoren systematisch mit dem hohen Anspruch, einen Überblick zu verschaffen. Die Anwendungserfahrungen der Autoren werden dabei sehr praxisnah und oft auch sehr detailliert dargeboten. Damit geraten die Autoren oftmals in Widerspruch zu ihrem Anliegen, Übersicht zu verschaffen. Vom interessierten Leser werden jedoch gerade diese Details mit Dank aufgenommen werden.
Der gewählte Schwerpunkt der Darstellungen widmet sich den niederen Schichten der Kommunikation und damit Problemen wie Signaldarstellung, Synchronisierung von Kommunikationsprozessen, Adressierung und Fehlersicherung. Dieser Band hilft allen Elektronikingenieuren und ist für die Hochschulausbildung in Informationstechnik/Informatik, Elektrotechnik/Elektronik und Automatisierungstechnik bestimmt.
Dr. I. Schreiber

Professor Schreiners UNIX-Sprechstunde

Technik, Tips und Tricks von awk bis yacc von Axel T. Schreiner, Carl Hanser Verlag, München, Wien, 1987, 316 Seiten, ISBN 3-446-14894-9

Die dem Betriebssystem UNIX zugeordnete Fachzeitschrift unix/mail aus der BRD hat auch in den Fachbibliotheken der DDR einen größeren Leserstamm. Der besondere Vorteil dieser Fachzeitschrift war, daß sie bereits 1983 versuchte, als erstes deutschsprachiges Periodikum umfassend über die UNIX-Anwendung und -Nutzung zu informieren. Dieser Anspruch wurde im Laufe der Jahre insofern immer weniger erfüllt, als daß der etwas akademische und elitäre Stil der unix/mail immer weniger in die sich auch kommerziell breit entwickelnde UNIX-Landschaft paßt.
Eine gleichbleibend über die Jahre besonders interessante Rubrik der unix/mail ist allerdings die von A. Schreiner gestaltete UNIX-Sprechstunde. Ihre Beiträge gesammelt, neu bearbeitet, zusammenhängend gegliedert und technisch auf den neuesten Stand gebracht (System V), wurden vom Hanser Verlag in Buchform vorgelegt. Ziel ist, für typische Probleme bei der UNIX-Anwendung möglichst elegante Lösungsvorschläge aufzubereiten, deren Resultate auch bei eigenen Problemstellungen weiterverwendet werden können. Das Spektrum der behandelten Themen reicht von einem kleinen Datenbankprogramm über einfache Dokumentationshilfen bis zu C-Programmiertricks und einem interaktiven Editor. Im einzelnen wird dabei eine außerordentlich große Palette von Themen (curses, grep, lex, yacc, make, man, ms, mm, nroff, u. v. a. m.) behandelt. Die Gefahr des Buches besteht besonders darin, daß ein großes Sammelsurium von für den Leser mehr oder weniger interessanten Einzelproblemen in fachlich exakter Weise abgehandelt wird. Es ist deshalb wichtig darauf zu achten, daß das Buch vor allem in die Hände der UNIX-Programmierer und Nutzer gehört, die schon größere Erfahrungen mit diesem System vorweisen können. Ein UNIX-Neuling wird von der Fülle der Detailinformationen und von der Lösungsvariantenvielfalt wohl immer erschlagen werden. *Dr. L. Claßen*

in Fokuso 2/88:

Gábor Witályos:
Bilder aus der Welt der Fraktale
(Computerkunst mittels Abbildung hochgradiger math. Funktionen, Vergrößerungen, Farbeffekte, 10 Abbildungen s/w, 1 in Farbe)
Dr. Peter Broczkó:
Produktion von Mikrocomputern im Jahre 1987 in den sozialistischen Ländern
(Hardware-Vergleiche, Tabelle der Eigenschaften, Vergleich des Angebots an Standardprogrammen)
Mark Fettes:
Das Mikrocomputer-Projekt Verteilte Sprachübersetzung – enorme Chance der Esperantolinguistik
(Stand der Erprobung des Projektes, einzige Konkurrenz: Japan)

Internationele Zeitschrift über Aktuelles in Wissenschaft und Technik, in Esperanto, beziehbar über: M. Behr, Koburger Str. 83, Markkleeberg, 7113 – Preis: 4 Hefte je 80 Seiten, pro Jahrgang 24,– M *B.*

Zeitschriftenschau

Cambridge Computer Z88

Einen weiteren Versuch, mit einem neuen Computer den Markt zu erobern, machte der „Vater" des gut bekannten ZX-Spectrum, Sir Clive Sinclair. Der Neue nennt sich Z88 und erinnert äußerlich nur noch geringfügig an den Spectrum. Produziert von der Firma Cambridge Computers Ltd., verfügt der Computer über ein Flüssigkristalldisplay sowie über einen wesentlich vergrößerten Speicherplatz und eine speicherresidente Programmbibliothek. Die Stromversorgung erfolgt durch Batterien (ca. 20 Stunden Betriebsmöglichkeit).
Ausgerüstet ist der Z88 mit einer in CMOS gefertigten Z80-CPU und einem Standard-RAM von 32 KByte sowie 128 KByte ROM. Zur Speichererweiterung sind RAM- und ROM-Module mit 32, 128 und 1024 KByte vorgesehen. Das Display läßt auf einer Zeile bis zu 106 Zeichen zu, 8 Zeilen sind gleichzeitig sichtbar. Für den Benutzer stehen pro Zeile jedoch nur 94 Zeichenplätze zur Verfügung. Die Tastatur gleicht im Aufbau der des Spectrums (Gummitaste, Folienkontakte).
Mit der Peripherie bzw. den Speichererweiterungen läßt sich der Z88 über eine RS-232-Schnittstelle bzw. über den Systembus verbinden. Der Betrieb am Netz ist durch ein externes Netzteil möglich.
Die Programmbibliothek umfaßt solche Nutzerprogramme wie Textverarbeitung, Kalkulations- und Datenbankprogramm sowie einen wissenschaftlich-technischen Rechner. Darüber hinaus läßt sich der Z88 noch als Notizkalender mit Alarmfunktion (Uhr ist eingebaut) und als Basic-Computer verwenden; benutzt wird ein BBC-Basic-Interpreter. Vorgesehen ist ein EPROM-Programmierzusatz, mit dem eigene Programme in die Module geschrieben werden können.
Programmroutinen zum Modembetrieb und zur Kopplung mit weiteren Computern sind ebenfalls im ROM enthalten.
Nicht vorgesehen sind der Betrieb an einem Monitor und der Anschluß eines Disketten- oder Kassettenlaufwerkes.

aus KOMPUTER (1988) 4, S. 17–18

Computer- und Medizindiagnostik

Das Ministerium für Gesundheitswesen der UdSSR hat ein computergestütztes Diagnostiksystem auf der Grundlage der Mikrorechner ISKRA-1256 und ISKRA-226 eingeführt. Gespeichert werden die Antworten und Ergebnisse der Fragen und Tests, die vom System über den Patienten erhalten wurden. Vom Computer werden Behandlungsvorschläge unterbreitet. Zur Kontrolle werden die Diagnosen parallel von Ärzten gestellt. Die Übereinstimmung soll über 90 % betragen.
Dieses Diagnosesystem hat besonders für die Reihenuntersuchungen in der Bevölkerung Bedeutung.

aus KOMPUTER (1988) 7, S. 8

Computer für Blinde und Sehschwache

Ein Problem dieses Personenkreises ist die beschränkte Zugriffsmöglichkeit auf visuelle Informationen (Druck und Fernsehen). Nur in beschränktem Maße sind solche Geräte, die Schrift in andere erfaßbare Informationen wandeln, nutzbar. Auch die Umsetzungsgeschwindigkeit des Zeichens in die Information war gering. Die Druckzeichen konnten in Vibrationen oder Töne umgesetzt werden.
Die Computertechnik macht es in zunehmendem Maße möglich, Text direkt in synthetische Sprache umzusetzen. Inzwischen sind auch Monitore, Tastaturen und Drucker für Brailleschrift vorgestellt worden. Gleichzeitig kann der Computer auch seine Kommunikation mit dem Bediener über Sprachsynthese abwickeln. Die Benutzung von Textverarbeitungssystemen in der Redaktionsarbeit ermöglicht das problemlose Erscheinen eines Textes in „normaler", und wenn die Druckmöglichkeit vorhanden ist, auch in Brailleschrift.
aus KOMPUTER (1988) 7, S. 12–13

Husky Hawk 8/16

Zu den sicherlich sehr seltenen Computertypen zählt der der Klasse der PCs zuzurechnende Husky Hawk 8/16. Es handelt sich dabei um einen im IBM-PC-Standard arbeitenden Computerzwerg mit einer Masse von nur 700 g. Das Gerät enthält zwei Module, die das Arbeiten unter CP/M (8 Bit) bzw. MS-DOS (16 Bit) gestatten. Das Gerät hat kein Diskettenlaufwerk und bezieht seine Programme über die RS-232-Schnittstelle und ein Modem. Zur Anzeige wird ein Flüssigkristallbildschirm genutzt. Nachfolgend zusammengefaßt die wichtigsten Daten des Computers:
• *Modul DOS:*
– 80C88-Prozessor
– RAM 640 KByte
– ROM 128 KByte
– 6,144 MHz Takt.
• *Modul CP/M:*
– HD-64B180-Prozessor
– RAM 352 KByte (davon 280 RAM-Disk)
– ROM 48 KByte Betriebssystem DEMOS (kompatibel zu CP/M)
– Basicinterpreter.
Tastatur 68 Mehrfunktionstasten
Bildschirm Flüssigkristall 8 Zeilen zu 40 Zeichen
Auflösung 240 × 64 Pixel
Interface 2 × RS 232 und Systembus.
aus KOMPUTER (1988) 8, S. 37–38

Turbo-Backup

Für die Herstellung der Arbeitskopien von Programmen oder Dateien in Betriebssystemen wie MS-DOS, PC-DOS u. ä. gibt es jetzt ein Kopierprogramm, das mit einer Geschwindigkeit von 0.95 MByte/min arbeitet. Falls der Computer mit einem DMA ausgerüstet ist, kann sich diese Geschwindigkeit noch auf ca. 3 MByte/min erhöhen. Damit ist der Nutzer kaum noch in der Lage, den manuellen Diskettenwechsel schnell genug zu vollziehen (für den Fall, daß von der Festplatte Backups hergestellt werden sollen). Anbieter des Programms ist DANSK DATA SUPPORT.
aus KOMPUTER (1988) 9, S. 32-34
Mi

MS-DOS-Version 4.01

Von Microsoft gibt es jetzt die zu IBMs PC-DOS 4.01 vollständig kompatible Version MS-DOS 4.01. Vorzüge dieser Version sind eine Bedienoberfläche mit der Möglichkeit, über grafische Eingabe Festplattendateien anzulegen, Applikationen zu starten und die Basis-DOS-Funktionen über Pulldown-Menüs und Dialogfenster auszuwählen. Die Unterstützung der Expanded Memory Specification (EMS) wurde verbessert. Ebenso sollen Probleme mit der Kompatibilität zu verschiedenen speicherresidenten Programmen ausgeräumt worden sein.
MP

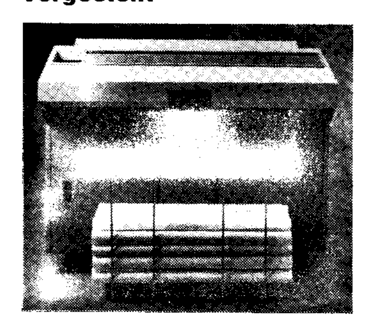
Leistungsstarker Universalrechner für kommerzielle Anwendungen

Die europäische Bull-Gruppe und die amerikanische Honeywell Bull Inc. kündigten mit dem System BULL DPS 9000 das neue „Flaggschiff" in ihrem gemeinsamen Großrechnerangebot an. Der Computer bietet mehr als die dreifache Leistung gegenüber dem bisherigen Topsystem DPS 90. Mit über 1000 Transaktionen pro Sekunde, 1000 MByte Hauptspeicherkapazität, 1000 KByte System-Pufferspeicher (Cache) und 1000 Ein-/Ausgabekanälen soll der DPS 9000 der derzeit leistungsfähigste kommerzielle Universalrechner auf dem Weltmarkt sein. (Die Leistungsmessung basiert auf dem TP 1/ET 1-Benchmark aus dem Bankanwendungsbereich.) Der neue Großrechner wird in vier Ausbaustufen angeboten. Die Modelle DPS 9000/91, / 92T, 93T und 94T („T" steht für voll redundante Hardwareausstattung) unterscheiden sich dabei im wesentlichen durch die Anzahl integrierter Zentralprozessoren.
Die Systemarchitektur des DPS 9000 umfaßt 1 bis 4 Zentralprozessoren (CPU), 1 bis 2 Systemsteuereinheiten (SCU), 1 bis 2 Hauptspeichereinheiten (MMU), 1 bis 4 Ein-/Ausgabeprozessoren (IOP), 1 bis 4 Vektorprozessoren (IVP) sowie integrierte Diagnoseprozessoren und Präventiveinrichtungen für höchste Systemsicherheit und -verfügbarkeit.
Die besonderen leistungsrelevanten Architekturmerkmale und Hardwarefunktionen sind eine doppelte simultan arbeitende Befehls- und Daten-Pipeline mit 5- bzw. 7stufiger Parallelverarbeitung zur Beschleunigung der Befehlsausführung, ein zweistufiges Hochgeschwindigkeits-Cache-Speicherkonzept, bestehend aus zwei 64-KByte-Zentralprozessor-Caches mit einem 512 KByte großen Cache in jeder Systemsteuereinheit zur Minimierung der Hauptspeicherzugriffe und somit zur Beschleunigung der zentralen Verarbeitungsprozesse, eine Hauptspeicherkapazität von max. 1024 MByte (1 Gigabyte) sowie ein virtueller Adreßraum von bis zu 4000 Gigabyte (4 Terabyte).
MP

Neues Open Access

Das integrierte Programm Open Access II wird es in einer kyrillischen Version geben. Das beabsichtigen die kalifornische Software Products International Inc. (SPI) und die sowjetische Akademie der Wissenschaften in Moskau. Für diese Aufgabe sei ein Joint-Venture geplant, einschließlich eines gemeinsamen Büros in Moskau. Mit dem Joint-Venture soll, so SPI, der Bedarf in der Sowjetunion an Standardsoftware gedeckt werden.
Open Access integriert bekanntlich Module für Kalkulation, Datenbank, Textverarbeitung, Grafik und Kommunikation.
Von der Münchner SPI GmbH wurde jetzt auch die Version 2.1 mit zusätzlichen Funktionen vorgestellt. Neu sind eine Rechtschreibhilfe und ein Trennprogramm für verschiedene Fremdsprachen sowie die Möglichkeit, verschiedene Applikationen einzubinden. Es lassen sich jetzt Grafikdateien erzeugen und extern weiterverarbeiten. Ebenso kann in der neuen Version direkt auf das Betriebssystem und auf EXE-Dateien zugegriffen werden.
MP

A0-Format-Laserplotter vorgestellt



Versatec, Hersteller elektrostatischer Farb- und Monochrom-Rasterplotter, bietet, nach eigenen Angaben, als erster Hersteller mit dem 8836 jetzt einen A0-Format-Laserplotter an. Merkmale sind hohe Plotgeschwindigkeit, Verwendung von nicht beschichtetem Papier, Ausgabe in Rollenform sowie Betrieb auch unter nicht optimalen Umgebungsbedingungen. Hauptsächlich werden CAD-Anwender in Mechanik, E-Technik, Architektur und Kartographie angesprochen, die auf einen Monatsdurchsatz von 2000 bis 3000 Plots im A0-Format kommen.
Bei einer Auflösung von 400 Punkten/Zoll beträgt die Plotgeschwindigkeit konstant 1 Zoll/s (2,54 cm/s). Zeichnungen im A0-Format werden in weniger als 70 s ausgegeben; das platzsparende Querformat (A1) dauert weniger als 1 min.
Ausgerichtet auf CAD-Anwender kann der VERSATEC 8836-Laserplotter mit den Standardschnittstellen RS232-C, Centronics oder Versatec Parallel Interface ausgestattet werden. Diese Schnittstellen ermöglichen es, Datenformate wie CalComp 906/907, CalComp 960 oder VERSATEC-Standard-Raster-Daten zu empfangen und auszuplotten. Optional sind direkter IBM-Kanal-Anschluß oder auch Ethernet-Anschluß möglich.
MP

Wordstar 5.0

Von Micropro gibt es jetzt die deutsche Version von Wordstar Professionell 5.0. Dieses erste laserdruckerfähige Wordstar verfügt über Layout-Funktionen und über Pull-down-Menüs, die dem Anwender alternativ zu den bisherigen Tastenkombinationen zur Verfügung stehen.
Im Modus *Advanced Page Preview* lassen sich die Layouts von maximal 144 Druckseiten überprüfen und die gewählten Schriften, Schriftattribute und Fontwechsel darstellen. Editiermöglichkeiten gibt es in diesem Bereich jedoch nicht. Korrekturen können nur im Textverarbeitungsmodus durchgeführt werden. Proportionalschrift wird aktiv unterstützt, so daß auch bei Fontwechseln oder Änderung der Druckattribute sogenannter feiner Blocksatz erhalten bleibt. Eine Seite kann in maximal acht Spalten umbrochen werden.
Wordstar 5.0 soll vor allem der täglichen Textverarbeitung dienen und im Bereich Textgestaltung dem Anwender neue Möglichkeiten zur Überprüfung des Layouts vor dem Druck eröffnen.
MP

MC 68040 angekündigt

Motorola hat für April 1989 die Vorstellung des neuen 32-Bit-Prozessors 68040 angekündigt. Er tritt die Nachfolge der 32-Bit-Prozessoren 68020 (1985) und 68030 (1987) an. Mit rund 1,2 Mio Transistoren, 15 MIPS (Millionen Instruktionen pro Sekunde) und 4 MFLOPS (Millionen Gleitkommaoperationen pro Sekunde) ist er mit dem von Intel angekündigten 80486 vergleichbar (siehe MP 12/88, Seite 384). Der erste 32-Bit-Prozessor von Intel ist der 80386 (1986), der mit 5 MIPS in der Rechnerleistung zwischen dem 68020 (3...4 MIPS) und dem 68030 (7 MIPS) liegt.
MP

Neue Chip-Fertigungsstraße bei IBM

Für die Chip-Massenproduktion nahm IBM Ende 1988 eine neue Fertigungsstraße in Betrieb, die Silizium-Wafer mit über 200 Millimeter Durchmesser (8 Zoll) verarbeiten kann.
Bis vor kurzem arbeitete IBM ausschließlich mit 5-Zoll-Wafern, aus denen bis zu 150 integrierte Schaltkreise gefertigt werden konnten. Mit seiner neuen Fertigungsstraße kann das Unternehmen in der gleichen Zeit bis zu 450 Chips herstellen. Die Linie arbeitet weitgehend automatisiert und dient unter anderem zur Produktion der 1-MBit-Chips von IBM.
ADN

RISC wird schneller

Tektronix und das Rensselaer Polytechnic Institute arbeiten zur Zeit an einem Computer mit RISC-Architektur, der mit 250 MIPS etwa die zehnfache Verarbeitungsgeschwindigkeit heutiger RISC-Systeme erreichen soll. Basis sind nicht Galliumarsenid-, sondern Silizium-ECL-ICs, die jedoch nach einem neuen Fertigungsverfahren hergestellt werden.
MP

Neue Chips von TI

Wissenschaftler von Texas Instruments berichteten auf einer Fachkonferenz der Elektronikbranche in San Francisco über die Entwicklung eines neuen integrierten Schaltkreises. Dieser Schaltkreis besteht aus *Quantum Tunneling Transistors* und soll die Basis für eine neue Generation von dünneren, schneller arbeitenden und kostengünstigeren Chips werden. Die *Quantum Tunneling Transistors* seien hundertmal dünner und tausendmal schneller als konventionelle Halbleiterbauelemente. Eine kommerzielle Fertigung der neuen Schaltkreisgeneration wird frühestens Mitte der 90er Jahre möglich sein.
MP

3500 PCs nach China

AST verkauft den chinesischen Ministerien für Wirtschaft und Kommunikation 3500 PCs für insgesamt 10 Millionen Dollar. Die Geräte, Premium/286 und 386, sollen mit Grafikkarten für die Darstellung chinesischer Schriftzeichen ausgerüstet werden. Dieser Vertrag ist einer der größten, die im Mikrocomputerbereich mit der Volkrepublik China abgeschlossen wurden.
MP

Projekt eines ultrakompakten Computers

Den Prototyp eines ultrakompakten Hochleistungscomputers soll die Firma Hughes Aircraft Co. fertiggestellt haben. Bei einem Volumen von $7{,}5 \times 7{,}5 \times 1{,}25\,cm^3$ soll der Rechner eine Arbeitsgeschwindigkeit von 10 Milliarden Operationen pro Sekunde erreichen. Langfristig soll das Entwicklungsziel in einem Rechner mit einer Geschwindigkeit von 100 Milliarden Operationen pro Sekunde bei einem Leistungsbedarf von weniger als 100 Milliwatt bestehen.
Die Kompaktheit des Rechners werde durch eine dreidimensionale Packungstechnik erreicht. Die Wafer werden übereinandergelegt und dann vertikal leitende Verbindungen hergestellt. Beim Prototyp wurden 32×32 Prozessorzellen je Wafer untergebracht und fünf dieser Arraywafer aufeinander gelegt. Die vertikalen Verbindungen sind das eigentliche Problem des Projektes.
Der Computer soll in Bordanlagen zum Einsatz kommen und Bildsignalverarbeitung in Echtzeit realisieren.

Quelle: Neue Zürcher Zeitung vom 19. 10. 1988
Wi

Flüssigkristallanzeige

Die bisher größte Flüssigkristallanzeige für Workstations und Personalcomputer wurde gemeinsam von den Firmen Toshiba und IBM entwickelt. Durch 1,58 Millionen Bildpunkte beim Schwarzweißbild (1440×1100) werden Bilder hoher Qualität und Auflösung erzielt. Beim Farbbild wird eine Auflösung von 700×550 Punkten erreicht. Mit 4 Punkten (rot, grün, blau und weiß) lassen sich 16 verschiedene Farben darstellen.
Fa

Klein und kraftvoll

In Japan wurde ein neuer Werkstoff auf der Grundlage von Neodym, Eisen und Bor entwickelt, der eine etwa 30 % höhere magnetische Leistungsfähigkeit aufweisen soll als die seit Mitte der siebziger Jahre verfügbaren Samarium-Kobalt-Legierungen. Die als Koerdym bezeichneten Magnete nehmen besonders in Japan einen schnell wachsenden Marktanteil ein. Von großem Vorteil ist dabei, daß die erforderlichen Rohstoffe besser verfügbar sind als Samarium und Kobalt.

Koerdym besteht zu 30 bis 34 % aus Neodym, zu 1,1 bis 1,3 % aus Bor, und der Rest ist Eisen. Die Legierung wird durch einen schmelzmetallurgischen Prozeß oder durch Koreduktion hergestellt. Danach wird das Material fein gemahlen, unter Einwirkung eines Magnetfeldes verpreßt, gesintert, wärmebehandelt und falls erforderlich mechanisch bearbeitet.

Der Nachteil des Koerdym besteht darin, daß zwar die Werte für Remanenz und Koerzitivfeldstärke außerordentlich hoch liegen, sie aber stärker temperaturabhängig sind als die der SmCo-Legierungen. Durch geeignete Werkstoffauswahl und Magnetkreisauslegung kann jedoch auch bei Arbeitstemperaturen von über 150 °C eine ausreichende Stabilität erreicht werden.

NdFeB-Magnete werden in Servomotoren für die Automatisierungstechnik und den Maschinenbau, die unter starkem Miniaturisierungsdruck stehenden Motoren für die Büro- und Datentechnik oder für Videogeräte und Kameras sowie Linearantriebe oder Drehmomentgeber angewendet. Immer häufiger erfolgen Anwendungen in magnetosensitiven Schaltern und Sensorsystemen oder in der Audiotechnik, beispielsweise in Kopfhörern oder Lautsprechern. Koerdym-Magnetteilchen bewirken zum Beispiel in Thermoplasten, daß dauermagnetische Bauteile in der üblichen Spritzgußtechnik hergestellt werden können.

Darüber hinaus sind die Koerdym-Magnete mit Werten bis zu 80 kJ/m^3 von besonderer Bedeutung, weil sie die Lücke zwischen dem magnetischen Niveau der gesinterten Hartferrite und dem der gesinterten Seltenerdmetall-Dauermagnete schließen könnten.

Quelle: Elektronik. – München 37 (1988) 23. – S. 24, 26 *Wi*

Ursachen für Chipausfälle

Nach Ansicht eines Forschungsteams der Cornell-Universität New York sind die unterschiedlichen Wärmeausdehnungskoeffizienten von Leitbahnen und Silizium die Ursache für die Ausfälle von fabrikneuen Chips. Untersuchungen sollen ergeben haben, daß eine Abhängigkeit zwischen Leitbahnbreite und Ausfallhäufigkeit besteht. Die zehnfache Verminderung der Leitbahnbreite soll zu einer 1000fachen Erhöhung der Ausfälle führen. Die Werkstoffexperten erklären die Ausfälle aus den mechanischen Spannungen, die während des Herstellungsprozesses entstehen.

Das im Verhältnis zu den Leitbahnen massive Silizium hindert die Aluminiumleitbahnen daran, beim Abkühlen die Spannungen vollständig abzubauen. Diese Spannungen bewirken, daß im Laufe der Zeit in den Metallbahnen kleinste Hohlräume entstehen. Durch den Abbau der inneren Spannungen schälen sich komplette, zusammenhängende Atomlagen ab, die sich dann in der Nähe wieder ablagern. Mit der Zeit entstehen so immer größere Hohlräume. Die Veränderungen sind nur abhängig von der Temperatur. Bei höheren Temperaturen kann sich der Prozeß innerhalb einiger hundert Stunden, bei Raumtemperatur innerhalb von ein bis zwei Jahren vollziehen – und das sogar bei unbenutzten Chips. Die gewonnenen Erkenntnisse über diese Vorgänge gestatten, Maßnahmen zur Veränderung zu untersuchen.

So gehen die Bemühungen in Richtung Einsatz bisher nicht benutzter Werkstoffe sowie auf die Realisierung flexibler Bindungen zwischen Leitbahnen und Siliziumsubstrat. Die weiter voranschreitende Miniaturisierung macht die Lösung dieses Problems erforderlich, wenn die Zuverlässigkeit der Chips weiter erhöht werden soll.

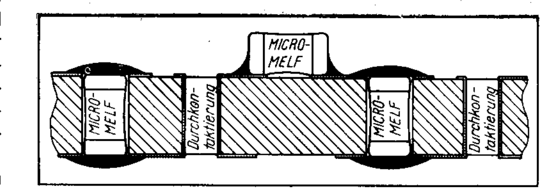
Quelle: Elektronik. – München 37 (1988) 22. – S. 30 *Wi.*

Nach SMD kommt IMD

Um den Platzbedarf von Bauelementen auf Leiterplatten noch weiter zu verringern, entwickelte die Firma Beyschlag eine neue Art von Widerständen, die als *Micro-MELFs* bezeichnet werden. Grundlage waren die zylindrischen SMD-Widerstände mit Metallkappen (MELFs), mit denen es durch eine Neukonstruktion möglich wurde, diese senkrecht auf der Leiterplatte anzuordnen und in die Durchkontaktierungen zu stecken. Diese Technik wird als *IMD*-Technik (Insert Mounted Device) bezeichnet. Nachdem diese Technik bis zur Serienreife entwickelt wurde, geht es nun darum, sie in der Massenbestükkung anzuwenden. Insbesondere wird noch nach einer Methode für die Fixierung vor dem Lötvorgang gesucht.

Die als kleinste passive Bauelemente bezeichneten Micro-MELFs können im eingelöteten Zustand, abhängig von der Bestückungsdichte der Platine und ihrer Gesamtbelastung, einer Belastbarkeit von bis zu 0,17 W ausgesetzt werden. Die Oberflächentemperatur darf bis 125 °C betragen. Mit 1 % Toleranz werden die Widerstände mit Werten von 22 Ω bis 220 kΩ angeboten, mit 2 % von 1 Ω bis 2,2 MΩ. Aufgrund ihrer Metallschichtausführung soll eine sehr gute Langzeitstabilität gegeben sein. Die Ausfallquote dieser Widerstände wird von Beyschlag mit weniger als 2×10^{-9} pro Stunde angegeben.

Quelle: Elektronik. – München 37 (1988) 22. – S. 78 *Fa*



Verfahren zur Gewinnung von Galliumarsenid

Die Vorteile von Galliumarsenid als Basismaterial für mikroelektronische Bauelemente sind seit längerer Zeit hinreichend bewiesen. Aufgrund des aufwendigen und umweltbelastenden Herstellungsprozesses konnte das Material bisher nicht das Silizium verdrängen.

Technologische Verfahren, die ohne den Hochtemperaturprozeß auskommen, bei dem arsenhaltige Gase entstehen, sind Forschungsgegenstand verschiedener Institutionen. Der Cornell-Universität New York soll es gelungen sein, eine Verbindung herzustellen, die Gallium und Arsen in dem erforderlichen Mischungsverhältnis enthält und aus der in gelöstem Zustand bei Raumtemperatur Galliumarsenid-Moleküle entstehen.

Das als Arsinogallan bezeichnete Material wird mit Butanol einer chemischen Reaktion ausgesetzt, bei der eine Chlorverbindung als Katalysator wirkt und bei der Galliumarsenid als rötliches Pulver anfällt. Die Chemiker sind der Auffassung, daß aus dem Arsinogallan die Fremdmaterialien ausgelöst werden und Galliumarsenid in Form von Einzelmolekülen entsteht, die sich später infolge ihrer Instabilität zu größeren Teilchen verbinden.

Mit einem ähnlichen Verfahren sollen Forscher in Ithaca Indiumphosphid gewonnen haben, das ebenfalls als Werkstoff in der Elektronik interessant ist.

Wann die Ergebnisse industriell genutzt werden können, ist noch nicht abzuschätzen.

Quelle: Elektronik. – München 37 (1988) 22. – S. 10 *Wi*

Röntgenlithographie zur Chipherstellung

Im Rahmen eines Langzeitforschungsprojektes zur Entwicklung und Auswertung neuer Lithographieverfahren arbeiten IBM-Wissenschaftler daran, Chips mit Hilfe der Röntgenstrahl-Lithographie herzustellen. Die Röntgenstrahl-Lithographie könnte in Zukunft die Massenproduktion von Speicherchips mit einer Kapazität von mehr als 64 MBit erlauben.

Mit Hilfe der Röntgenstrahl-Lithographie können sehr viel engere Schaltkreismuster gestaltet werden als mit der Ultraviolett- und der Lichtlithographie. Röntgenstrahlen haben eine wesentlich kleinere Wellenlänge. Da die Wellenlänge die Auflösung bestimmt, lassen sich um so feinere Strukturen abbilden, je kleiner sie ist. Die Röntgenstrahl-Lithographie könnte außerdem die Fehlerquote in der Produktion erheblich herabsetzen, da Staubpartikel, die bei Licht normalerweise zu Mängeln führen, von Röntgenstrahlen durchdrungen werden.

Den IBM-Wissenschaftlern soll es gelungen sein, einen ganzen Speicherchip mit Strukturbreiten von 0,5 μm herzustellen. Zu den wesentlichen Bestandteilen des neuen Verfahrens gehören:

– ein gerasterter Spiegel zur Verteilung der Röntgenstrahlen über die zu belichtende Fläche
– ein Berylliumfenster, um das Vakuum des Elektronen-Speicherrings von der Belichtungskammer zu trennen
– vier Röntgenmasken
– ein Stepper, um jede Maske und den Wafer zu halten und zurückzuführen
– ein Masken-/Wafer-Ausrichtungssystem.

Während des Belichtungsvorgangs werden Röntgenstrahlen vom Speicherring auf den Spiegel geleitet. Dieser bewirkt eine gleichmäßige Beleuchtung der Maske. Das etwa 20 μm dicke Berylliumfenster trennt die heliumgekühlte Belichtungskammer vom Vakuum und filtert die Röntgenstrahlen aus dem Strahlenspektrum des Speicherrings heraus. Der Wafer wird etwa 40 μm von der Maske entfernt in Position gebracht, und ein Roboter bringt Maske und Wafer in Schritten von 0,02 μm in die richtige Stellung. Die Belichtung erzeugt im Fotolack des Wafers Muster von weniger als 0,5 μm Breite. Die nicht benötigten Röntgenstrahlen werden von der mit Gold belegten Maske zurückgehalten. Um auf dem Wafer fünf je ein Quadratzoll große Felder zu belichten, muß dieses Verfahren fünfmal wiederholt werden. Nach weiteren Prozeßschritten wird mit den anderen drei Masken in gleicher Weise verfahren. Als Ergebnis entsteht ein funktionsfähiger Speicherchip.

Quelle: Elektronik. – München 37 (1988) 20. – S. 16 *Fa*

100 ns schnelle 1-MBit-EPROMs

Die von der Wafer Scale Integration entwickelten 1-MBit-EPROMs sind in Keramik-DIP oder -LCC-Gehäusen untergebracht und erreichen durchschnittliche Zugriffszeiten von 100, 120 und 150 ns. Die Firma arbeitet gegenwärtig daran, eine verbesserte Version mit 55 ns zu entwickeln, die bereits in diesem Jahr in kleinen Stückzahlen produziert werden soll. Außerdem werden von der Firma 512- und 256-KBit-EPROMs produziert, die Zugriffszeiten von 90 ns aufweisen.

Quelle: Elektronik. – München 37 (1988) 22. – S. 7 *Fa*

Speicherchip mit Supraleiter

Einen 1-KBit-Chip mit einer Zugriffszeit von 570 Picosekunden soll die japanische Firma NEC entwickelt haben.

Durch die Verwendung von supraleitendem Material wurde eine zehnmal höhere Geschwindigkeit gegenüber herkömmlichen Chips erreicht. Damit wäre erstmals der Übergang von einem supraleitenden Einzelkontakt zu einem hochintegrierten Bauelement gelungen. *Wi.*



Bericht

DECWORLD '88

Die bisher größte Computerausstellung der Firma Digital Equipment Corporation (DEC), die DECWORLD '88, fand vom 12.–23. September 1988 im Festival- und Kongreßpalast in Cannes/Frankreich statt. Für die Mitglieder der Anwenderorganisation DECUS, die ihr 24. Europäisches Symposium unmittelbar vor der Ausstellung abhielt, war die Ausstellung bereits am 8. und 9. September geöffnet. Insgesamt zählte die DECWORLD '88 ca. 17000 Gäste aus mehr als 60 Ländern. Im Tiefgeschoß des Festival- und Kongreßpalastes wurde auf über 10000 m^2 beeindruckend das auf vielen Gebieten führende Know-How der Firma in Hard- und Software demonstriert. Die Ausstellung stand unter dem Motto „Integrating the Enterprise" und hatte das Ziel, komplexe integrierte Lösungen zur Automatisierung aller Bereiche eines modernen Produktionsbetriebes zu zeigen. Dementsprechend war die Ausstellung in verschiedene Fachbereiche eingeteilt, wie

- Manufacturing (CAM)
- Production Environment (CIM)
- Solution Development Environment (rechnergestützte Softwareentwicklung)
- End-User Computing and Office Integration (Arbeitsplatzrechner und Büroautomatisierung)
- Enterprise Networking (verteilte Netzwerke für große Firmen)
- System and Network Management (Rechnersystem- und Netzwerkverwaltung)
- Enterprise Services (Computeranwendungen bei Firmendienstleistungen)
- Financial Services (Automatisierung des Finanz- und Bankwesens)
- Service Industries (Automatisierung in der Dienstleistungsindustrie)
- Telecommunication Industries (Computeranwendungen in der Nachrichtenübermittlung).

Damit trat der Computer auf der Ausstellung gegenüber der Demonstration von kompletten Anwendungslösungen in den Hintergrund. Trotzdem wurde das gesamte Know-How in der Computertechnologie, vor allem im Zentralteil der Ausstellung, dem Technologiezentrum, aber auch in zahlreiche Anwendungslösungen integriert gezeigt. Als rechentechnische Basis wurden etwa 50 Mini- und Superminirechner und über 250 Workstations mit einer Gesamtrechenleistung von über 1400 MIPS (Millionen Befehle pro Sekunde) installiert.
Die Vernetzung aller Computer und Workstations erfolgt über zwei Ethernetsysteme, die durch einen Router miteinander gekoppelt waren. Über den Nachrichtensatelliten TELECOM1 wurde eine Verbindung nach Reading, dem Hauptsitz der englischen Tochtergesellschaft, und von dort über den Nachrichtensatelliten IBS nach Boston und zu den amerikanischen Technologiezentren der Firma hergestellt.
Aus der Vielzahl der auf der DECWORLD '88 gezeigten Produkte und Anwendungslösungen seien hier im Bericht nur einige wenige Entwicklungen aus dem Bereich der Computertechnologie herausgegriffen.
Zur Verwaltung und Überwachung eines großen Rechnernetzes wurde im Mittelpunkt des Technologiezentrums auf der Ausstellung eine **Netzwerküberwachungszentrale** gezeigt, deren Hauptaufgaben die Leistungsüberwachung und -planung von großen Netzwerken, Überwachung der Netzwerksicherheit, Fehlerdiagnostik in großen Netzwerken, Online-Diagnostik von angeschlossenen Rechnern und die Überwachung des Netzwerkverkehrs sind.
Die Netzwerküberwachungszentrale ist mit einem speziellen Serviceprozessor auf der Basis einer Micro VAX II implementiert. (Bild 8; alle Bilder siehe 4. Umschlagseite).
Auf der Basis eines PBX-**Terminal-Servers** wurde die **Verkopplung eines PBX-Netzes** (ISDN-Netzes) mit einem Ethernetsystem vorgeführt. Zur Kopplung des PBX-Terminal-Servers mit der PBX-Zentrale wird das S2-Interface nach ECMA mit 2,048 MBit/s eingesetzt, das 30 parallele Datenübertragungen gestattet. Der Anschluß des ISDN-Telefons an die PBX-Zentrale erfolgt mit dem S1-Interface, das einen 64-KBit/s-Kanal für die digitale Telefonie und einen 64-KBit/s-Kanal für eine Datenübertragung (Terminalanschluß) zur Verfügung stellt. (Bild 1) Es kamen ISDN-Produkte verschiedener Hersteller zum Einsatz.
Zur Integration von PCs in die VAX-Umgebung wurde die Personal Computing Architecture (PCSA) mit einer **MicroVAX-2000 als VAX/VMS-Server** mit der MS-DOS-Server-Software vorgestellt. An einem Server können 10–15 PCs mit dem Betriebssystem MS-DOS über ein Thin-Wire-Ethernet-Kabel (10 MBit/s, max. 150 m Gesamtlänge) angeschlossen werden.
Demonstriert wurde ein gemischtes PC-Netz mit IBM PS/2 Mod. 30 unter PC-DOS und COMPAQ sowie VAX-mate (Intel 80286) mit MS-DOS. Auf den PC-DOS bzw. MS-DOS-Rechnern läuft die Client-Software DECnet-PCSA. Für den IBM PS/2 Mod. 30 wurde der Thin-Wire-Ethernet-Adapter DEPCA angeboten. Neben der Terminalemulation und Filetransferdiensten wurden integrierte Anwendungen mit gemeinsam genutzten Datenbasen vorgeführt.
Im Technologiezentrum war eine Hochleistungsrechenstation mit mehreren VAX-Rechnern der Typenreihe **VAX-88xx** und den neuesten Anlagen im mittleren Leistungsbereich VAX-62xx installiert. Die VAX-8810 ... 8840 verfügen über 1 bis 4 Prozessoren zu je 6 MIPS. Die neue Betriebssystemversion VAX/VMS V5.0 unterstützt das symmetrische Multiprocessing, das die nahezu volle Ausschöpfung der nominalen Prozessorleistung gestattet. Damit verfügt die VAX-8840 über 24 MIPS.
Die Typenreihe VAX-6210 ... 6240, auf der Basis des MicroVAX-III-Chipsatzes, wurde erst im Sommer 1988 angekündigt. Wie die MicroVAX-3xxx-Produkte, die die gleichen Chips benutzen, hat ein Prozessorelement eine Leistung von 2,7 MIPS.
Die **VAX-6240** (Bild 2) hat mit 4 Prozessorkarten im symmetrischen Multiprocessing eine Leistung von ca. 11 MIPS. Vorteile der VAX-62xx sind der interne Hochleistungsbus mit 100 MByte/s, der max. Speicherausbau bis 256 MByte auf Basis der 1-MBit-Chips und max. 6 E/A-Kanäle auf der Basis des VAXBI-Busses. Damit beträgt die maximale E/A-Transferrate über 60 MByte/s. Der VAXBI-Bus gestattet den Anschluß von Ethernet-LAN, VAX-Clustern, lokalen Plattenspeichersubsystemen und beliebiger sonstiger Peripherie. Die VAX-62xx-Modelle mit den verbesserten Leistungsparametern werden zum Standardsystem im mittleren Leistungsbereich, wobei der E/A-Durchsatz wesentlich höher als bei den VAX-88xx-Modellen ist, die ihren Vorteil in der erheblich höheren Prozessorleistung haben.
Im unteren Leistungsbereich befinden sich die MicroVAX-Produkte **MicroVAX-2000** und **VAXstation 2000** (Bild 3), die keinen für den Anwender zugänglichen Standardbus haben. Bei einer Leistung von nahezu 1 MIPS (Mikroprozessor 78032) repräsentieren sie auch das untere Preisniveau.
Die Q-Bus-kompatiblen MicroVAX-II-Produkte auf der Basis des gleichen Mikroprozessors sind nun durch **MicroVAX-3xxx-Produkte** (Bild 4 und 5) ersetzt worden, die den wesentlich leistungsfähigeren Mikroprozessor 78034 mit 2,7 MIPS nutzen. Durch Beibehaltung des internen Q-Busses sind diese Systeme für den Anwender relativ gut frei konfigurierbar.
Die DEC-Workstation-Produkte (VAXstation 2000 bzw. VAXstation 3xxx) basieren auf entsprechenden MicroVAX-Produkten, die anstelle des Standardterminals einen Videoadapter mit Videomonitor VR260 (schwarzweiß) oder VR290 (color) verwenden. Die Videomonitore haben bei 19"-Bildschirmdiagonale eine Auflösung von 1024 × 864 Pixel. Zusätzlich können eine Maus und/oder ein Digitalisiertablett zum Einsatz kommen. Weiterhin ist ein Ethernet-Adapter für den LAN-Anschluß obligatorischer Bestandteil.
Im Technologiezentrum kamen auch die neuen Plattenspeicher RA 70 und RA 90 zum Einsatz. Das **RA 70-Winchesterlaufwerk** auf der Basis der 5¼"-Technologie mit 280 MByte und 27 ms Zugriffszeit findet vorwiegend in MicroVAX- und VAXstation-Produkten Verwendung. Das **RA 90-Winchesterlaufwerk** mit einer Kapazität von 1,2 GByte und 25 ms Zugriffszeit in der 9"-Technologie ist zweifellos eine Spitzenleistung. Bemerkenswert ist vor allem, daß damit auch DEC die 14"-Technologie (RA 81 – 464 MByte, RA 82 – 622 MByte) endgültig verlassen hat. Das RA 90 dient vorrangig als externer Massenspeicher bei den größeren VAX-Systemen. Mit 8 Laufwerken konfiguriert, hat das **Speichersubsystem SA 600** eine Kapazität von 10 GByte.
In einem VAX-Cluster können an einem hierarchischen Speichercontroller HSC 70 max. 6 SA 600 eingesetzt werden. Gegenüber den Fortschritten in der Plattenspeichertechnologie ist die Backup-Technologie bei DEC deutlich zurückgeblieben. Nach wie vor hat hier das 1/2"-Industriestand-Magnetband mit einer Aufzeichnungsdichte von 1600 bzw. 6250 bpi seinen festen Platz. Für die Micro-VAX-Produkte mit RA 70-Laufwerken ist aber das neue Kassettenmagnet-bandgerät TK 70 mit 296 MByte eine akzeptable Lösung.
Auf der DECWORLD '88 wurde auch erstmals ein **optischer Plattenspeicher RV 20** (Bild 6) nach der WORM-Technologie (Write Once – Read Multiple) mit 2 GByte von DEC vorgestellt. Die 12"-Platte wird doppelseitig beschrieben und ist auswechselbar (Kassettenplatte). Die Lebensdauer der Aufzeichnungen wird mit mindestens 30 Jahren angegeben. Die Kapazität einer Platte entspricht etwa 55–60 Magnetbandspulen (6250 bpi). Das RV 20 ist in erster Linie als Archivspeicher geeignet. Als Backup-Medium sind die Datenträgerkosten zu hoch, jedoch sind gegenüber Magnetbandgeräten die Betriebskosten wesentlich niedriger. Gegenüber den magnetischen Winchesterplatten-speichern ist das RV 20 etwa 10mal langsamer.
Für lokale Netzwerke wurde der **Printserver 40** (Bild 7) als monochromatisches Druckersubsystem vorgestellt. Der Printserver wird über ein Ether-net-Interface direkt an das LAN angeschlossen. Die Druckleistung beträgt etwa 40 A4-Kopien pro Minute, was ungefähr 1,6 Millionen Kopien pro Monat entspricht. Die Papierformate sind von A5 bis A3 wählbar. Die Auflösung beträgt etwa 12 × 12 Bildpunkte je mm^2. Auf der Grundlage von POSTSCRIPT können über den eingebauten Rastergrafikprozessor Text, Grafik und natürliche Bilder gemischt werden. Resident sind 29 Schriftarten (Fonts) vorhanden, weitere Fonts können bei Bedarf geladen werden. Auf dem Software-Sektor war der fortschreitende UNIX-Einsatz auf Workstations ein zentrales Thema. Grundlage der weiteren Entwicklung sind die POSIX-Standards und die OSF-Produkte (Open Software Foundation). Das Betriebssystem ULTRIX-32, Version 3.0, ist das erste UNIX, das zu folgenden Standards bzw. Industriestandards kompatibel ist:
IEEE POSIX P1003.1 (am 22.8.88 bestätigt)
OSF Base Level 0 Spezifikation
Xwindow System Version 11 (Release 2)
System-V-Interfacedefinition, SVID Release II
X/OPEN Portability Guide.
OSF ist eine neue mit Kapital von DEC, IBM, Apollo Domain u. a. gegründete Firma, die durch OSF-Produkte die Kompatibilität von Softwa[re] reproduzten unter den UNIX-Betriebssystemen der Firmen IBM, DEC, Bull, Siemens, Nixdorf usw. s[i]chern will.
Dazu werden von OSF die entsprechenden Produkte über Lizenzen aufgekauft und OSF-gerecht weiterentwickelt. Damit stellt OSF eine Ergänzung zu den POSIX-Standards da[r] und sichert auch die Kompatibilität z[u] BSD-UNIX-Systemen.

Prof. Dr. Thomas Ho[...]

SET DELIMITER TO DEFAULT Definieren einer Zeichenkette, die als Begrenzer für Eingabebereiche benutzt wird (Standard ":").
SET DELIMITER ON|OFF ON schaltet die Benutzung der durch SET DELIMITER TO definierten Zeichenkette ein.
SET DEVICE TO SCREEN|PRINTER Die Ausgaben der @-Kommandos werden auf dem Bildschirm oder dem Drucker ausgegeben.
SET ECHO ON|OFF Das Echo der Kommandozeilen erscheint auf dem Bildschirm (ON).
SET ESCAPE ON|OFF Das Drücken der <ESC>-Taste unterbricht (ON) die Abarbeitung eines Kommandofiles.
SET EXACT ON|OFF Zeichenketten unterschiedlicher Länge liefern beim Vergleich immer FALSE (ON).
SET FILTER TO [<bedingung>]
Die Bedingung dient als Auswahlbedingung für alle Kommandos.
SET FIXED ON|OFF Alle Dezimalzahlen werden mit einer festen Anzahl an Dezimalstellen angezeigt (ON).
SET FUNCTION <taste> **TO** <zeichenkette> Definition von Funktionstasten.
SET FORMAT TO [<filename>]
Ein Formatfile (.fmt) für die Eingabe im Direktmodus wird aktiviert und bestimmt bei allen nachfolgenden Eingaben im Direktmodus die Bildschirmmaske.
SET HEADING ON|OFF Feldbezeichner werden als Überschriften ausgegeben.
SET HELP ON|OFF Unterstützung bei Fehlern durch Ausschriften (ON).
SET INDEX TO [<fileliste>] Eröffnen der aufgeführten Indexfiles.
SET INTENSITY ON|OFF Die Eingabebereiche bei der Eingabe im Direktmodus werden invers dargestellt (ON).
SET MARGIN TO <n> Definition des linken Randes auf dem Drucker.
SET MENUS ON|OFF Hilfsmenüs werden bei Operationen im Direktmodus angezeigt (ON).
SET PATH TO [<pfad>] Definition eines Suchpfades.
SET PRINT ON|OFF Ausgabe auf den Drucker (außer @-Ausgaben).
SET PROCEDURE TO [<proz_file>] Eröffnen eines Prozedurfiles.
SET RELATION TO [<schlüssel>] **INTO** <alias>
Verbinden von zwei Datenbankfiles über einen Schlüssel.
SET SAFETY ON|OFF Das Überschreiben eines vorhandenen Files führt zu einer Anfrage an den Nutzer (ON).
SET STEP ON|OFF Schrittweise Abarbeitung eines Programms (ON).
SET TALK ON|OFF Das Ergebnis der Ausführung von Kommandos wird auf dem Bildschirm ausgegeben (ON).
SET UNIQUE ON|OFF Das erste (ON) oder alle Datensätze mit gleichem Schlüssel werden beim Indizieren in die Indexfiles aufgenommen.
SKIP [<n>] Der Dateizeiger wird um n Datensätze vorwärts (n > O) oder rückwärts (n < O) bewegt (Standard n = 1).
SORT TO <neues_files> **ON** <feld$_1$> [/A] [/D] [,<feld$_2$>] [/A] [/D] ...] [<gb>] [FOR <bedingung>]
Das aktive Datenbankfile wird nach den angegebenen Feldern und der angegebenen Reihenfolge sortiert.
STORE <ausdr> **TO** <speichervar_liste>
<speichervar> = <ausdr>
Speichern des Wertes des Ausdrucks in alle angegebenen Speichervariablen.
SUM [<gb>] <ausdr_liste> [**TO** <speichervar_liste>] [**FOR|WHILE** <bedingung>] Berechnen der Summen der in der Ausdrucksliste angegebenen Werte für alle ausgewählten Datensätze.

TEXT <text>
ENDTEXT Ausgabe des Textes aus einem Kommandofile ohne Makroersetzung.
TOTAL TO <filename> **ON** <schlüssel> [<gb>] [[**FIELDS**] <feldliste>] [**FOR|WHILE** <bedingung>]
Komprimieren eines Datenbankfiles, wobei alle Sätze mit gleichem Schlüssel zu einem Satz zusammengefaßt werden. Die in der Feldliste aufgeführten numerischen Felder werden addiert.
TYPE <filename> [**TO PRINT**] Gibt ein Textfile auf dem Bildschirm aus.

UPDATE [RANDOM] ON <schlüssel> **FROM** <alias>
REPLACE <feld$_1$> **WITH** <ausdr$_1$> [,<feld$_2$> **WITH** <ausdr$_2$> ...]
Korrektur eines Datenbankfiles, wobei die Korrekturdaten von einem anderen File eingelesen werden.
USE [<filename>] [**INDEX** <fileliste>] [**ALIAS** <alias>]
Das Datenbankfile <filename> wird mit den in der <fileliste> aufgeführten Indexdateien in Benutzung genommen.

WAIT [<prompt>] [**TO** <speichervar>]
Die Abarbeitung des Programms wird unterbrochen, bis eine Taste gedrückt wird. Deren Zeichenkode wird der Speichervariablen zugewiesen.

ZAP Löschen aller Datensätze des aktiven Datenbankfiles.

Funktionen und ihre Typen

<c_ausdr> Ausdruck vom Typ String (character)
<d_ausdr> Ausdruck vom Typ Datum
<n_ausdr> Ausdruck vom Typ numerisch

Funktion und Aufruf	Ergebnistyp	Bemerkung
&<speichervar>	String	Makroersetzung
ASC(<c_ausdr>)	numerisch	num. Wert eines Zeichens
AT(<c_ausdr1>,<c_ausdr_2>)	numerisch	Position von <c_ausdr$_1$> in <c_ausdr$_2$>
BOF( )	logisch	Fileanfang
CDOW(<d_ausdr>)	String	Wochentag
CHR(<n_ausdr>)	String	Zeichen mit dem Wert <n_ausdr>
CMONTH(<d_ausdr>)	String	Name des Monats
COL( )	numerisch	akt. Spalte auf dem Bildschirm
CTOD(<c_ausdr>)	Datum	Konvertierung String in Datum
DATE( )	Datum	aktuelles Datum
DAY(<d_ausdr>)	numerisch	Tag des Monats
DELETED( )	logisch	Abfrage der Löschmarkierung
DOW(<d_ausdr>)	numerisch	Nummer des Wochentags
DTOC(<d_ausdr>)	String	Konvertierung Datum in String
EXP(<n_ausdr>)	numerisch	Exponentialfunktion
FILE(<c_ausdr>)	logisch	Existenz eines Files
INKEY( )	numerisch	Kode einer Taste
INT(<n_ausdr>)	numerisch	ganzzahliger Teil
LEN(<c_ausdr>)	numerisch	Länge eines Strings
LOG(<c_ausdr>)	numerisch	Logarithmus
LOWER(<c_ausdr>)	String	Konvertierung in kleine Buchstaben
MONTH(<d_ausdr>)	numerisch	Nummer des Monats
PCOL( )	numerisch	Spaltenposition des Druckkopfes
PROW( )	numerisch	Zeilenposition des Druckkopfes
RECNO( )	numerisch	Nummer des aktuellen Satzes
ROUND(<c_ausdr>,<dez>)	numerisch	Runden
ROW( )	numerisch	akt. Zeile auf dem Bildschirm
SPACE(<n_ausdr>)	String	<n_ausdr> Leerzeichen
SQRT(<n_ausdr>)	numerisch	Quadratwurzel
STR(<n_ausdr>,<länge>,[<dez>])	String	Konvertierung eines Wertes in einen String
SUBSTR(<c_ausdr$_1$>,<start>[,<länge>])	String	Substring
TIME( )	String	Systemzeit
TRIM(<c_ausdr>)	String	Wegschneiden von Leerzeichen am Ende
TYPE(<c_ausdr>)	String	Typ eines Bezeichners
UPPER(<c_ausdr>)	String	Konvertierung in große Buchstaben
VAL(<c_ausdr>)	numerisch	Konvertierung String in numerischen Wert
YEAR(<d_ausdr>)	numerisch	Jahreszahl

Kursorsteuerung im Direktmodus

Taste	Ersatz	Wirkung
↑	**^E**	Eine Zeile oder ein Feld nach oben oder in BROWSE zum vorherigen Satz
↓	**^X**	Eine Zeile oder ein Feld nach unten oder in BROWSE zum nächsten Satz
←	**^S**	Ein Zeichen nach links. Ist durch ^Home ein zusätzliches Menü eingeschaltet, so Auswahl
→	**^D**	Ein Zeichen nach rechts. Ist durch ^Home ein zusätzliches Menü eingeschaltet, so Auswahl
^←	**^Z**	In REPORT: verschiebt den Bildschirmausschnitt nach links. in MODIFY COMMAND: zum linken Rand
^→	**^B**	In REPORT: verschiebt den Bildschirmausschnitt nach rechts. In MODIFY COMMAND: zum Zeilenende
<—		Löscht das Zeichen links vom Kursor
Del	**^G**	Löscht das Zeichen über dem Kursor
End	**^F**	Ein Wort nach rechts. In BROWSE: ein Feld nach rechts
^End	**^W**	Abspeichern aller Änderungen und Verlassen des Direktmodus. Bem.: Verlassen wir in EDIT den Gültigkeitsbereich, so ist die Wirkung wie bei ^End
Esc	**^Q**	Verlassen des Direktmodus. Die Änderungen werden verworfen und der Zustand vor der Korrektur wird wiederhergestellt. In APPEND und BROWSE: Nur die Änderungen im letzten Satz werden verworfen. Die anderen sind gültig
Home	**^A**	Ein Wort nach links. In BROWSE: Ein Feld nach links
^Home	**^]**	In BROWSE, LABEL, REPORT: Schaltet ein zusätzliches Menü ein und aus. In EDIT, CHANGE, APPEND, wenn der Kursor in einem MEMO-Feld steht: Aufruf des Editors für die Bearbeitung von Memo-Texten. Rückkehr mit ^End
Ins	**^V**	Schaltet den Einfügemodus ein und aus
PgUp	**^R**	In BROWSE: Blättern rückwärts
PgDn	**^C**	In BROWSE: Blättern vorwärts
Enter	**^M**	Zum nächsten Feld. In APPEND als erstes Zeichen im ersten Feld: Beenden von APPEND. In Edit im letzten Feld des letzten Satzes: wie ^End. In MODIFY COMMAND, wenn der Einfügemodus eingeschaltet ist: Zeile einfügen. In Menüs: Auswahl einer Option
	^N	In MODIFY STRUCTURE: Einfügen eines Feldes. In MODIFY COMMAND: Einfügen einer Zeile
	^T	Löscht das Wort rechts vom Kursor
	^U	Schaltet die Löschmarkierung für einen Satz ein und aus. In MODIFY REPORT, MODIFY STRUCTURE: Löscht die Felddefinition
	^Y	Löscht den Inhalt des Feldes. In MODIFY COMMAND: Löscht eine Zeile
	^KR	In MODIFY COMMAND: Fügt ein File an der Kursorposition ein
	^KW	In MODIFY COMMAND: Schreibt den gesamten Text in ein File

Außerdem wirken einige Steuerzeichen auch auf der Kommandoebene von dBASE:

←	**^H**	Löscht das Zeichen links vom Kursor
Enter	**^M**	Beendet eine Kommandozeile
	^P	Schaltet den Drucker logisch ein und aus (hard copy)
	^S	Stoppt die Bildschirmausgabe. Diese kann durch Drücken einer beliebigen Taste fortgesetzt werden
	^X	Löscht die Kommandozeile

zusammengestellt von Dr. Wilfried Grafik

ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 3
S. 65–96





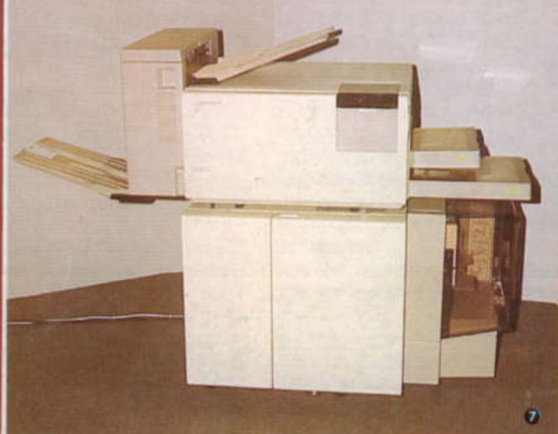








Bilder von der DECWORLD '88

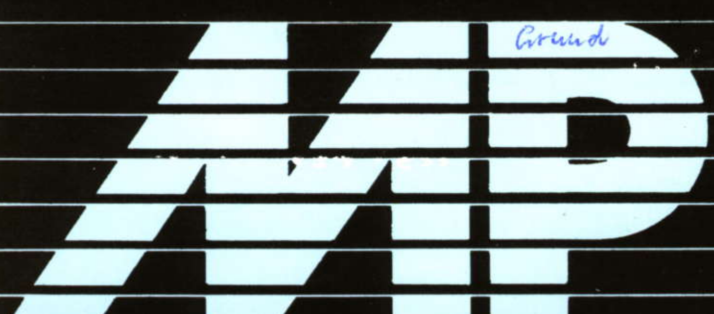




❶ **Terminal VT 340 am ISDN-Telefon**
OPUS-4000 (64 kBit/s) der Fa. Thomson CSF
❷ **VAX 6240**
4 Prozessoren mit Mikroprozessor 78034, FPU 78134, 11 MIPS, max. 256 MByte, max. 60 MByte/s E/A-Transferrate, Ethernet, Computerinterconnect
❸ **VAXstation 2000**
Mikroprozessor 78032, 4 oder 6 MByte, 19"-Bildschirm (1024 × 864 Pixel), 71- oder 159-MByte-Festplatte, 95-MByte-Kassettenmagnetband, Thin-Wire-Ethernet
❹ **VAXstation 3200**
Mikroprozessor 78034, FPU 78134, 8 MByte, 19"-Bildschirm (1024 × 864 Pixel), 159-MByte-Festplatte, 296-MByte-Kassettenmagnetband, Ethernet
❺ **VAXstation 3500**
Mikroprozessor 78034, FPU 78134, 16 MByte, 19"-Bildschirm (1024 × 864 Pixel), 280-MByte-Festplatte, Ethernet
❻ **Optischer Plattenspeicher RV 20**
WORM (Write-once Read multiple), Wechselplatte 2 GByte
❼ **Printserver 40**
11,8 Pixel/mm, 40 Seiten/min, 29 Schriften, Laserdrucker mit Trockentoner, Ethernet
❽ **Serviceprozessor**
auf Basis von MicroVAX-II mit Remote Services Console zur Überwachung großer Netzwerke

Lesen Sie dazu den Bericht in diesem Heft

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK



Heft 4 · 1989

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0232-2892

Architekturen von Supercomputern

mit Referenzkarte dBase III Plus

Neuerscheinungen

Lichtwellenleitertechnik

Von Prof. Dr. sc. techn. Erhard Grimm und Prof. Dr.-Ing. habil. Walter Nowak. 328 Seiten, 332 Bilder, 24 Tafeln, Leinen, DDR 32,– M, Ausland 46,– DM. Bestellangaben: 553 736 2/Grimm, Lichtwellen.

Als Lehrbuch konzipiert. Neben den Grundlagen und den Berechnungsverfahren für Lichtwellenleiter werden die optoelektronischen Wandler (Strahlungsquellen und -empfänger) behandelt. Weiterhin sind Fragen der Systemgestaltung und die Meßtechnik Gegenstand des Buches.

Auch für Leser aus der Praxis geeignet, die sich tiefgründig mit diesem neuen Gebiet beschäftigen wollen.



Computer-Englisch

Zusammengestellt von Dipl.-Ing. Richard Krischak und Dipl.-Päd. Ursula Schöne. Etwa 64 Seiten, Broschur, DDR etwa 3,80 M, Ausland etwa 8,– DM. Erscheint in Kürze. Bestellangaben: 554 141 7/Krischak, Computerenglisch

Alphabetisch geordnetes englisch-deutsches Spezialwörterverzeichnis als Übersetzungshilfe für alle Dokumentationen in englischer Sprache auf dem Gebiet der Computertechnik, besonders für Bedieneranleitungen sowie Software.

Diskrete Optimierungsmodelle

Effektive Algorithmen und näherungsweise Lösung

Von Prof. Dr. sc. nat. Knut Richter, Dr. rer. nat. Peter Bachmann und Dr. rer. nat. Stephan Dempe. 188 Seiten, 41 Bilder, 12 Tafeln, Kunstleder, DDR 26,– M, Ausland 36,– DM. Bestellangaben: 553 902 6/Richter, Optimierung

Gegeben werden anwendungsreife effektive Algorithmen für so wichtige Belange wie die optimale Aufteilung der Jahresproduktion auf Monate unter Berücksichtigung des schwankenden Bedarfs, die optimale „Echtzeit"-Steuerung integrierter Fertigungsabschnitte und den optimalen Einsatz von Robotern.



Rechnergestützte Qualitätssicherung

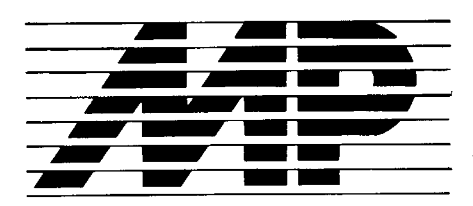
Von Prof. Dr.-Ing. habil. Dietrich Hofmann. 288 Seiten, 124 Tafeln, Leinen, DDR 30,– M, Ausland 40,– DM. Bestellangaben: 553 744 2/Hofmann, Qualitätssicher.

Fach- und Handbuch für Ingenieure aller Industriezweige zum Einarbeiten in die Grundlagen und praktischen Anwendungen der rechnergestützten Qualitätssicherung CAQ.

Auslieferung durch den Fachbuchhandel

VEB VERLAG TECHNIK BERLIN

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 4 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titel Tatjana Stephanowitz

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard-Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

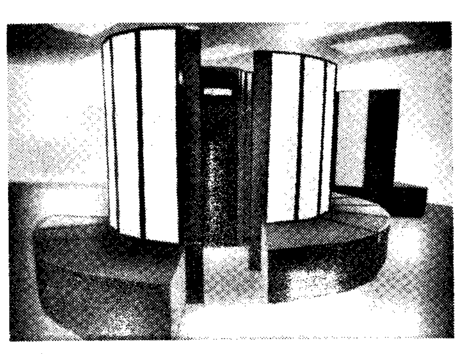
Redaktionsschluß: 10. Februar 1989

AN (EDV) 49837

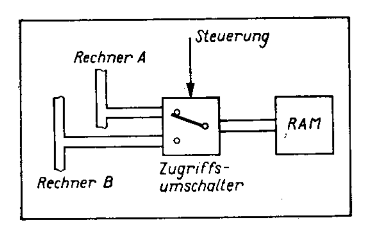
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Auf der **Seite 99** finden Sie den Beitrag „Hochleistungsrechner mit Mikroprozessoren". Er behandelt Minisupercomputer bzw. Superminicomputer und geht näher auf die Hypercube-Topologie als eine **Architektur von Supercomputern** ein.



Ein Dual-Port-RAM ist das Thema des Artikels auf der **Seite 102.** Die Steuerung der zeitlichen Abläufe erfolgt durch einen getakteten Automaten.

Vorschau

Mikroprozessorsystem 80600:
- CPU 80601
- Buscontroller 80606
- Fehlererkennungs- und -korrekturschaltkreis 80608
- DRAM-Controller 80610
- SCC 82530 und CIO 82536

Inhalt

Computer für Kinder





Die *Anwendung von Computern bei der Herausbildung von speziellen Denkfertigkeiten bei Kindern und Schülern* war das Thema einer Ausstellung im November 1988 im Haus der Sowjetischen Wissenschaft und Kultur in Berlin. Drei verschiedene Rechnersysteme mit entsprechender Software konnten von den Besuchern selbst erprobt werden.
Eine originelle Lösung zur Anwendung für Vorschulkinder im Alter von 5 bis 7 Jahren ist der Spiel-Rechner-Komplex auf der Basis der sowjetischen Heimcomputer **Elektronika-BK-0010** in Verbindung mit Farbfernsehgeräten (Bild 1). Die Konfiguration wurde in Zusammenarbeit von Pädagogen, Psychologen, Ärzten, Designern und Computerfachleuten entwickelt. Für die standardgemäße Folientastatur gibt es speziell an die Programme angepaßte und für die Kinder leicht erfaßbare Aufleger, die ein leichtes Bedienen ermöglichen. Von einem zentralen Arbeitsplatz aus können über 30 verschiedene Programme geladen werden. Das System umfaßt bis zu 10 Plätze und eine komplette didaktisch-methodisch durchdachte Spielumgebung. Kinder können spielerisch an den Umgang mit Computern herangeführt werden. Neben der Herausbildung von Fähigkeiten und Fertigkeiten wird die psychologische Bereitschaft zur künftigen Tätigkeit in einer hochautomatisierten Wirtschaft entwickelt.
Für den Einsatz in der Schul- und Berufsausbildung wurde der Rechnerkomplex **KORVET** aus Baku vorgestellt (Bild 2). Der Lehrerarbeitsplatz mit 8080-Rechner hat hier folgende Daten: 64 KByte Arbeitsspeicher, 144 KByte RAM-Disk, 24 KByte Basic-ROM, 48 KByte Grafikspeicher, Bildschirm mit 512 × 256 Punkten und 16 aus 256 Farben, 2 Diskettenlaufwerke 1 MByte (unformatiert), CP/M-kompatibles Betriebssystem. Er läßt sich mit bis zu 15 Schülerarbeitsplätzen koppeln, die den gleichen Rechner, nur mit Magnetbandkassettenspeicher, enthalten. Auf der Basis dieses Systems bietet die Kooperative *Radiostrojenije* Baku eine Reihe interessanter Softwarelösungen an. Die Schüler können weitgehend selbständig arbeiten. Dem Lehrer gelingt es leicht, den Lernprozeß zu führen und zu kontrollieren. Durch Daten- und Programmtransfer von den Massenspeichern des Lehrerarbeitsplatzes wird ein zeitlich ökonomisches Arbeiten möglich.
Als Einzelarbeitsplatz zum Aneignen von Wissen und zum Trainieren von Fertigkeiten auf verschiedenen Gebieten wurde der Personalcomputer EC 1841 mit K 1810 WM86-CPU und MS-DOS-kompatiblem Betriebssystem aus sowjetischer Fertigung vorgestellt. Es ist zu erwarten, daß Maschinen dieser Klasse mit ihren universellen Möglichkeiten gerade auch für viele Bereiche der Ausbildung in der Sowjetunion und den anderen sozialistischen Staaten eine hohe Bedeutung erhalten werden.

Matthias Fischer

Positive Bilanz 1988

Wie sich aus der Mitteilung der Staatlichen Zentralverwaltung für Statistik über die Durchführung des Volkswirtschaftsplanes 1988 ergibt, hat sich in Verwirklichung der ökonomischen Strategie der SED das hohe Tempo der Einführung von Schlüsseltechnologien in der DDR fortgesetzt. So wurden – nach vorläufigen Angaben – beispielsweise folgende Ergebnisse erzielt:

- Wesentlich ausgebaut wurde die mikroelektronische Basis der Volkswirtschaft. Mit den ersten in der DDR hergestellten Mustern des 1-MBit-Speicherschaltkreises auf der Basis eigener technologischer Verfahren sind Grundlagen geschaffen worden, um die Produktion in diesem Jahr aufzunehmen. Die Produktion von 64-KBit-Speicherschaltkreisen ist 1988 erhöht worden. Mit der Fertigung von 256-KBit-Speicherschaltkreisen wurde begonnen.
Gegenüber dem Jahr 1987 wuchs die Produktion von monolithisch-integrierten Schaltkreisen um 57 Prozent, Büro- und Personalcomputern um 16 Prozent, Speichern für die Computertechnik um 95 Prozent, Druckern für Computer um 33 Prozent, technologischen Spezialausrüstungen für die Herstellung von aktiven Halbleiterbauelementen um 20 Prozent, mikrolithografischen Geräten um 28 Prozent.
- Die rechentechnische Basis ist mit der Produktion von 57366 Büro- und Personalcomputern weiter gestärkt worden. Wesentlich gesteigert wurde die Fertigung von 16-Bit-Computern mit größerer Rechengeschwindigkeit und erhöhter Speicherkapazität.
- Gegenwärtig sind in der Volkswirtschaft rund 70000 CAD/CAM-Arbeitsstationen und -Systeme im Einsatz; die Anzahl der eingesetzten Industrieroboter wuchs auf insgesamt über 90000.
- Mit dem Ausbau der Mikroelektronik werden in immer stärkerem Maße Möglichkeiten zur Steigerung der Effektivität und Leistungsentwicklung erschlossen. Nahezu die Hälfte der Produktion der metallverarbeitenden Industrie wurde von der Mikroelektronik bestimmt. *MP*

Neue Fachsektion der Gesellschaft für Informatik der DDR konstituiert

Am 1. und 2. Dezember 1988 fand in Suhl die erste Jahrestagung der neuen Fachsektion „Informatik und Gesellschaft" der Gesellschaft für Informatik der DDR zum Thema „Computer und Gesellschaft" statt.
Die Fachsektion stellt sich die Aufgabe, neue wissenschaftliche Erkenntnisse, die sich auf die Wechselbeziehungen zwischen Informatik und Gesellschaft beziehen, einem breiten Kreis von Interessenten zugänglich zu machen. Sie fördert kooperatives Zusammenwirken von Institutionen und Wissenschaftlern, unterstützt den Austausch von Forschungsergebnissen und den wissenschaftlichen Meinungsstreit zu gesellschaftlichen Voraussetzungen und Konsequenzen neuer Informations- und Kommunikationstechnologien sowie zu Konzepten und Methoden ihrer bewußten Gestaltung und erarbeitet dazu eigene Standpunkte. Maßgebend für das Erreichen der mit der gesellschaftlichen Nutzung der Informatik angestrebten politischen, ökonomischen und sozialen Ziele ist die Beherrschung ihres gesellschaftlichen Umfeldes. Wirtschaftswissenschaftliche, organisatorische, arbeitswissenschaftliche, leitungsseitige, soziologische und juristische Aspekte sind dabei ebenso im Blickfeld wie der Bezug zur Weltanschauung, zu globalen Problemen und zur kulturellen Entwicklung.
Das Leitungsgremium besteht aus *Prof. Dr. sc. oec. Wolfgang Uhr*, TU Dresden, Sektion Sozialistische Betriebswirtschaft, als Vorsitzenden der Fachsektion, *Prof. Dr. phil. habil. Klaus Fuchs-Kittowski*, Humboldt-Universität zu Berlin, Sektion Wissenschaftstheorie und -organisation, als Stellvertretenden Vorsitzenden, *Prof. Dr. oec. habil. Gerd Friedrich*, Stellvertretender Direktor des Zentralinstituts für Sozialistische Wirtschaftsführung beim ZK der SED, Berlin, sowie den Leitern der Arbeitsgruppen und *Dr. oec. Dipl. Math. Horst Bürdek*, TU Dresden, als Sekretär der Fachsektion. Die Fachsektion besteht aus folgenden sechs Arbeitsgruppen (AG):
AG 1: Rechnerunterstützte Betriebswirtschaft/Informatik und Wirtschaftswissenschaften (Leitung: *Prof. Dr. sc. oec. Wolfgang Uhr*, TU Dresden, Sektion Sozialistische Betriebswirtschaft)
AG 2: Informatik und Recht (Leitung: *Prof. Dr. sc. jur. R. Osterland*, TU Dresden)
AG 3: Komplexe nutzerorientierte Gestaltung von Informationssystemen und Mensch-Maschine-Beziehung (Leitung: *Prof. Dr. phil. habil. Klaus Fuchs-Kittowski*, Humboldt-Universität zu Berlin, Sektion Wissenschaftstheorie und -organisation)
AG 4: Informatik und Weltanschauung (Leitung: *Prof. Dr. habil. L. Striebing*, TU Dresden)
AG 5: Informatik und Kultur (Leitung: *Prof. Dr. habil. H. Völz*, AdW der DDR)
AG 6: Informatik und globale Probleme (Leitung: *Prof. Dr. sc. phil. Dr. Ing. M. Roth*, TH Ilmenau) *H.W.*

GaAs-Kristalle aus Freiberg

In einer 1988 von der Bergakademie Freiberg und dem VEB Spurenmetalle Freiberg geschaffenen Versuchsanlage werden seit Dezember 1988 Galliumarsenid-Kristalle von höchster Qualität gezüchtet. Die Entwicklung dieses Basismaterials für die Mikroelektronik in Partnerschaft mit der Industrie gehört zur erfolgreichen Forschungsbilanz der Freiberger Hochschule im vergangenen Jahr.
Neben dem Galliumarsenid wurden auch Vorprodukte für hochfeine Sondertonerden entwickelt. Sie sind Ausgangsprodukt bei der Herstellung neuer keramischer Werkstoffe. Beispielsweise wurde für die Mikroelektronik und andere Bereiche ein dispersionsverfestigter Aluminiumoxid-Werkstoff geschaffen. *ADN*

Hochleistungsrechner mit Mikroprozessoren

Dr. Hans Heuer
Technische Universität Magdeburg, Sektion Informatik

Dr. Hans Heuer (53) studierte von 1954–1959 Mathematik an der Universität Rostock. Nach Abschluß des Studiums war er als wissenschaftlicher Mitarbeiter im VEB Wissenschaftlich-Technisches Zentrum Ausrüstungen für die Schwerindustrie und Getriebebau Magdeburg in der Forschung beschäftigt.
1968 nahm er eine Tätigkeit als Leiter Programmierung an der heutigen Technischen Universität „Otto von Guericke" Magdeburg auf. Gegenwärtig ist er wissenschaftlicher Oberassistent an der Sektion Informatik.
Er arbeitet zur Zeit an der Untersuchung von Möglichkeiten zur Erhöhung der Effektivität durch den Einsatz neuartiger Rechnersysteme, an der Erforschung aktueller Strukturen von Rechnern mit Parallelverarbeitung und unterschiedlicher Topologien von Verbindungsnetzwerken, an der Bewertung von Programmiersprachen und der Entwicklung von Algorithmen für Parallelrechner, sowie an der praktischen Lösung von Aufgaben der Anwendung.

Einleitung

Das Spektrum der Rechner war bisher am unteren Ende durch die Mikrorechner und am oberen Ende durch die sogenannten Supercomputer gekennzeichnet. Die Leistung solcher Höchstleistungsrechner wie Cray-1, Cray-2, Cray X-MP (siehe Bild 1), Cyber 205, Cyberplus, ETA 10 (siehe 4. Umschlagseite), NEC SX-2 oder Fujitsu VP200 beträgt mehrere hundert MFLOPS (1 MFLOPS = 1 Million Gleitpunktoperationen je Sekunde). Auf Grund der hohen Kosten (Kaufpreis ab 5 Mio Dollar aufwärts) und der speziellen Architektur kann ein effektiver Einsatz solcher Supercomputer im allgemeinen nur für eine relativ geringe Anzahl ausgewählter Aufgaben erfolgen. Dies bedingt ihre Konzentration in Rechenzentren. Mit den Höchstleistungsrechnern ETA 10 mit den Modellen ETA 10-P, ETA 10-Q, ETA 10-E und ETA 10-G zeichnet sich jetzt eine Dezentralisierungstendenz ab /1/.
Außerdem hat die stürmische Entwicklung der Mikroprozessortechnik in Verbindung mit dem Übergang von der traditionellen Struktur zu neuen Architekturkonzepten zur Entwicklung einer großen Vielfalt neuer hochleistungsfähiger Rechnersysteme geführt. Die Rechner des oberen Bereiches benutzen solche wichtigen Elemente der Parallelverarbeitung wie Pipelines, mehrfache Funktionseinheiten und Multiprozessortechniken. Hinzu kommen große physische Speicher bis 4 GByte, große schnelle Caches bis 1 MByte und hohe Transfergeschwindigkeiten. Bezüglich ihrer Verarbeitungsleistung führen diese Rechner nahe an die Supercomputer heran. Sie unterscheiden sich von ihnen vor allem durch die erheblich niedrigeren Kosten /2/, /3/, /4/.

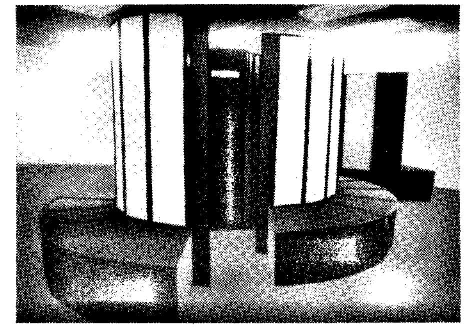


Bild 1 Supercomputer X-MP 48 von Cray

Minisuper und Supermini

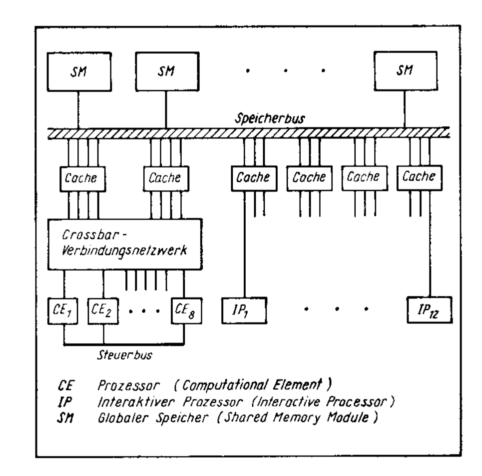
Die Verfügbarkeit von 16- und 32-Bit-Hochleistungsmikroprozessoren wie Intel 80286/80386, Motorola 68010/68020/68030 NS 16032/32032 zusammen mit ihren Gleitpunkt-Koprozessoren bietet die Möglichkeit des Aufbaus von Hochleistungsrechnern aus relativ preiswerten Komponenten.
Die Hochleistungsrechner auf der Basis der modernen Mikroprozessortechnik werden entweder als Minisuper-Computer (Minisuper) oder als Supermini-Computer (Supermini) bezeichnet, je nachdem, ob sie eine unmittelbare Vektorverarbeitung ermöglichen oder nicht. Die Minisuper, die auch in Anlehnung an die Vektorrechner der Firma Cray Research als *Crayettes* bezeichnet werden, stellen auf Grund der Fähigkeit der Vektorverarbeitung Geräte dar, die vor allem für Vektoren großer Länge effektiv arbeiten. Sie sind jedoch so entworfen, daß sie darüber hinaus auch die Lösung einer Vielzahl anderer Aufgaben ermöglichen, beispielsweise Büroautomatisierung oder Verwaltung großer Datenbanken. Allgemein ist in dieser Leistungsklasse ein Verschwimmen der Unterschiede zwischen Skalar- und Vektorrechnern zu beobachten und damit auch ein Verwischen der Unterschiede zwischen Standalone-Rechner und Zusatz-Rechner. Die neuesten Rechner sind Universalrechner, die in ihrem architektonischen Entwurf mehr einer VAX als einer Cray ähnlich sind.
Die Vektorverarbeitung der Minisuper-Computer wird an die mehr universellen Anwendungen angepaßt, das heißt an die kommerziellen Anwendungen. Die für eine effektive Arbeit erforderlichen Vektorlängen werden kürzer, die Verarbeitungseinheiten kleiner, aber zahlreicher. Hierdurch werden nicht nur die Kosten verringert, sondern es steigt auch die Flexibilität bei der Ausführung von Skalaroperationen.
Die aktuelle Situation ist gekennzeichnet durch ein Vordringen der Minisuper-Computer in die Richtung der Supercomputer, während gleichzeitig die Supermini-Computer die Leistung von traditionellen Großrechnern erreichen /3/, /4/.
In Tafel 1 sind Merkmale einiger Minisuper-Computer bzw. Supermini-Computer angegeben. Bei der Bewertung der Produktivität ist zu beachten, daß die tatsächliche Leistung nicht nur durch die Hardware bestimmt ist, sondern in wesentlichem Maße auch von der zu lösenden Aufgabe abhängt.

Tafel 1 Wichtige Merkmale der Minisuper-Computer C-1, SCS-40, FX-8 und der Supermini-Computer Connection Machine und T-Series

Rechner (Firma)	Anzahl der Prozessoren	Maximale Produktivität in MFLOPS
C-1 (Convex Computer Corp. of Richardson, Texas)	1	60
SCS-40 (Scientific Computer Systems of San Diego, California	1	40
FX-8 (Alliant of Acton, Mass.)	8	60
Connection Machine (Thinking Machines Corp. of Cambridge, Mass.)	16384, 65536	1000
T-Series (Floating Point Systems, Beaverton, Ore.)	16...16384	abhängig von der Anzahl der Prozessoren

Convex hat mit der C-1 für ihren Prozessor wesentliche Anleihen bei der Cray-Architektur gemacht. Für den Softwareentwickler sieht der Rechner wie eine VAX aus. Der Convex-Fortran-Compiler akzeptiert Programme, die für die Abarbeitung unter dem VAX-Betriebssystem geschrieben wurden. Der Rechner SCS-40 von Scientific Computer Systems wurde so entworfen, daß er Cray-Programme unverändert verarbeitet. Die Ausführung erfolgt zwar nicht ganz so schnell wie auf einer Cray, aber dafür wesentlich kostengünstiger. Mehrere Supercomputernutzer haben diesen Rechner installiert und benutzen ihn für die Entwicklung und das Tuning von Programmen für ihre Cray-Rechner. Der Rechner FX-8 verfügt über 8 Prozessoren, die zusammen eine Verarbeitungsleistung von 60 MFLOPS ermöglichen. Ein spezieller Steuerbus dient der Synchronisation. Bild 2 zeigt die Architektur dieses Multiprozessorsystems. Die Connection Machine hat 16384 oder 65536 1-Bit-Prozessoren, die in einem 14- oder 16dimensionalen Hypercube angeordnet sind. Dieser Rechner wurde vor allem für Aufgaben der Künstlichen Intelligenz entwickelt. Die Rechner der T-Reihe sind als 4- bis 14dimensionale Hypercubes mit 16 bis 16384 Prozessoren geplant. Als Prozessoren werden die von der Firma Inmos entwickelten Transputer benutzt. Dementsprechend erfolgt die Pro-

Bild 2 Architektur des Alliant FX-8



grammierung in der problemorientierten Sprache OCCAM, der Muttersprache der Transputer. Sie ermöglicht das Schreiben effektiver paralleler Programme.

Experimentalsysteme

Neben den industriell hergestellten Hochleistungsrechnern auf der Basis der Mikroprozessortechnik gibt es eine Vielfalt von Experimentalsystemen, bei denen zunächst nicht der Leistungsaspekt im Vordergrund steht. Häufig erfolgt die Entwicklung solcher Systeme an Universitäten und Hochschulen. Diese Systeme sind insofern von Interesse, als sie Hinweise auf mögliche neue Entwicklungsrichtungen geben.
An der University of Southern California wurde mit dem TX16 ein experimentelles Multimikroprozessorsystem mit geteiltem (globalem) Speicher entwickelt. Seine Architektur zeigen die Bilder 3, 4 und 5. TX16 ist ein Cluster aus 16 Transputern und wurde entworfen als Prozessor für verarbeitungsin-

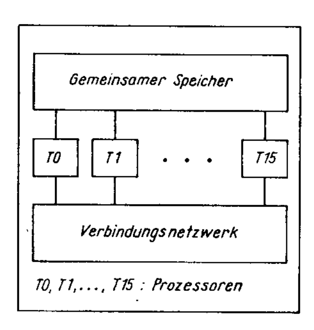


Bild 3 Architektur des TX16, Gesamtarchitektur

Topologie des Verbindungsnetzwerks

Bei Rechnern mit mehreren Prozessoren spielt die Struktur des Verbindungsnetzwerks, das die Prozessoren untereinander bzw. die Prozessoren mit dem gemeinsamen Speicher verbindet, eine ganz wesentliche Rolle für die Leistungsfähigkeit des Gesamtsystems. Eine sehr populäre Architektur für Hochleistungsrechner auf der Basis von Mikroprozessoren, aber auch für Rechner geringerer Leistung, ist die Hypercube-Topologie. Seit 1983 wurden bereits mehr als 100 unterschiedliche Hypercubes entwickelt /9/, /10/, /11/, /12/.
Bei diesen Systemen sind die einzelnen Prozessoren bzw. Verarbeitungselemente, die als Knoten bezeichnet werden, wie die Eckpunkte eines n-dimensionalen Würfels (n-Würfel) angeordnet und durch Kommunikationswege zwischen den Knoten, die den Kanten des Würfels entsprechen, miteinander verbunden. Ein n-Würfel besitzt 2^n Knoten, von denen jeder mit den n nächsten Knoten verbunden ist. Ein Knoten kann mit Knoten, mit denen er nicht unmittelbar verbunden ist, ebenfalls kommunizieren, indem die Informationen über geeignete Zwischenknoten weitergeleitet werden. Die durchschnittliche Anzahl der Teilstrecken ist dabei nur gleich n/2. Die Hypercube-Topologie hat als wesentliche Vorteile die relativ geringen Kosten bei relativ geringer durchschnittlicher Anzahl der Teilstrecken für die Kommunikation zwischen zwei beliebigen Knoten, ein mit der Dimension n ständig größer werdendes Verhältnis der Kommunikationsmöglichkeiten santen Verarbeitungsleistung auch als Superhypercubes bezeichnet.
Daneben gibt es eine Vielzahl weiterer Hypercube-Rechner mit einem breiten Spektrum der Verarbeitungsleistung, zum Beispiel Mark II, Mark III, iPSC, iPSC-VX, NCube/ten /4/, /11/, /12/.

Abbildung des Problems auf die Hardware

Eine wesentliche Aufgabe bei der Nutzung der Hochleistungsrechner ist die Abbildung des Problems auf die Hardware. So hat beispielsweise die Ausführung von sogenannten Vektoroperationen nur einen Sinn, wenn die Vektoren eine bestimmte Mindestlänge besitzen, das heißt eine Länge, für die eine Vektoroperation weniger Zeit beansprucht als eine entsprechende Anzahl von Skalaroperationen. Bei der Anwendung von Iterationsverfahren zur Lösung eines Problems in einem ebenen Bereich ist ein Verbindungsnetzwerk mit Gitterstruktur besonders geeignet, während für Suchprozesse eine Baumstruktur vorteilhaft ist. Für Aufgaben wechselnden Charakters sind Rechner mit flexibler Struktur besonders geeignet. Ein Vorteil der besprochenen Hypercube-Computer besteht darin, daß ihr flexibles Verbindungsnetzwerk ermöglicht, verschiedene Topologien für verschiedene Anwendungen zu wählen. Durch geeignete Festlegung der Verbindungen entstehen solche Strukturen wie Ring, Baum, mehrdimensionales Gitter oder aber Hypercube.
Die Bilder 7 und 8 zeigen dies am Beispiel von zwei solchen Strukturen.

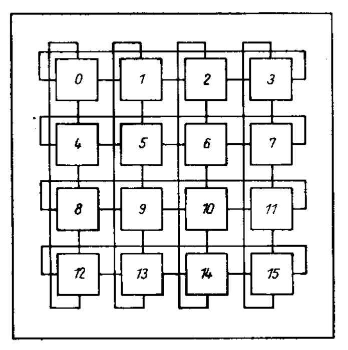
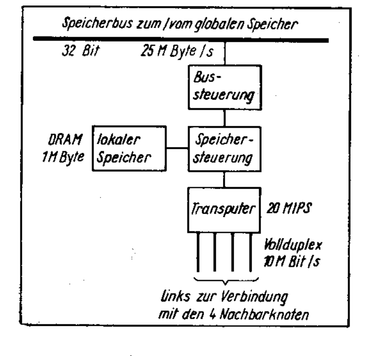


Bild 4 Architektur des TX16
Die Verbindung der Prozessoren wurde in Form eines Gitters ähnlich wie beim Rechner ILLIAC IV realisiert. Jeder der 16 Knoten des Gitters entspricht einem Prozessor mit lokalem Speicher.

Bild 5 Architektur des TX16
Darstellung eines einzelnen Knotens mit dem Transputer als Verarbeitungs- und Kommunikationseinheit und einem lokalen Speicher. Über die 4 Links des Transputers erfolgt die Verbindung mit den 4 Nachbarknoten. Über den Bus ist ein Zugriff auf den lokalen Speicher jedes beliebigen Knotens möglich. Damit stellt der TX16 ein Beispiel für eine Architektur mit verteiltem globalen Speicher dar.



tensive Aufgaben. Es ist vorgesehen, mehrere TX16 über eine Bus-Hierarchie zu verbinden, um große Multiprozessorsysteme zu bilden /5/. Das Multiprozessorsystem M^5PS (Modulares Multi-Mode-Multi-Mikroprozessor-System) der Rheinisch-Westfälischen Technischen Hochschule Aachen hat als wichtiges Ziel, ein modulares Rechnerkonzept zu entwickeln, dessen Leistung stufenweise den Anforderungen der unterschiedlichen Nutzer angepaßt werden kann. Ein Cluster oder Teilsystem ist ein homogenes symmetrisches Multiprozessorsystem, das bis zu 8 Mikroprozessoren umfaßt, die über einen gemeinsamen Bus und über einen gemeinsamen Speicher miteinander kommunizieren. Das Erscheinen von Systemen auf dem Markt, die im Grundprinzip mit dem System M^5PS vergleichbar sind (Encore Multimax mit bis zu 10 Prozessoren, ELXSI System 6400 mit bis zu 10 Prozessoren), bestätigen die Richtigkeit der gewählten Linie /6/, /7/, /8/.

zur Verarbeitungsleistung sowie die Möglichkeit der leichten Erweiterbarkeit. Bild 6 demonstriert die Erweiterung anhand der Kombination zweier 2dimensionaler Hypercubes zu einem 3dimensionalen Hypercube sowie durch die Zusammensetzung von zwei 3dimensionalen Hypercubes zu einem 4dimensionalen Hypercube. Der bekannte Cosmic-Cube, der am California Institute of Technology entwickelt wurde, gilt als der Prototyp der Hypercube-Topologie. Er besitzt 64 Prozessoren, die zu einem 6dimensionalen Würfel vereinigt wurden. Jeder Eckpunkt des Würfels ist ein einzelner Rechner auf der Basis des Intel-8086/8087-Chipsets mit 128 KByte RAM. Die Programmierung erfolgt in einer der bekannten Programmiersprachen (Fortran, Pascal, C u. a.), die um Anweisungen für die Kommunikation erweitert wurden. Eine solche Hypercube-Topologie liegt der bereits erwähnten T-Serie zugrunde. Ebenso besitzt die Connection Machine eine Hypercube-Verbindungsstruktur. Die beiden letzten Rechner werden auf Grund ihrer impo-

Anforderungen an Software und Algorithmen

Für die volle Ausschöpfung der potentiellen Leistung der Hochleistungsrechner auf der Basis der Mikroprozessortechnik hat offensichtlich auch die Software eine wesentliche Bedeutung. Die Betriebssysteme und die Programmiersysteme (Programmiersprachen und Compiler) müssen der Rechnerarchitektur angepaßt sein. Das gilt entsprechend für die Anwendungsprogramme. Darüber hinaus müssen auch die Algorithmen der Hardware entsprechen /13/.
Die Erfahrung mit den Minisuper-Computern zeigt, daß die Nutzung konventioneller Programmiersprachen, die um Anweisungen für die Synchronisation und Kommunikation erweitert wurden, nicht besonders günstig ist. Bezüglich der mit dem Einsatz der Transputer verbundenen Sprache OCCAM gibt es noch keine umfassende Bewertung. Es kann jedoch bereits jetzt eingeschätzt werden, daß OCCAM eine ausdrucksstarke Programmier-

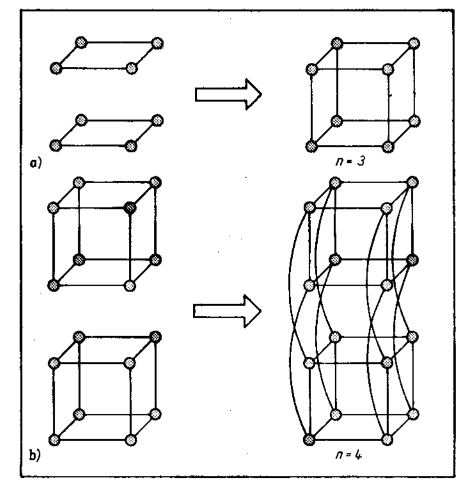


Bild 6 Hypercube-Topologie
a) Erweiterung eines 2dimensionalen Hypercubes durch Kopplung mit einem weiteren 2dimensionalen Hypercube zu einem 3dimensionalen Hypercube.
b) Verbindung von zwei 3dimensionalen Hypercubes zu einem 4dimensionalen Hypercube.

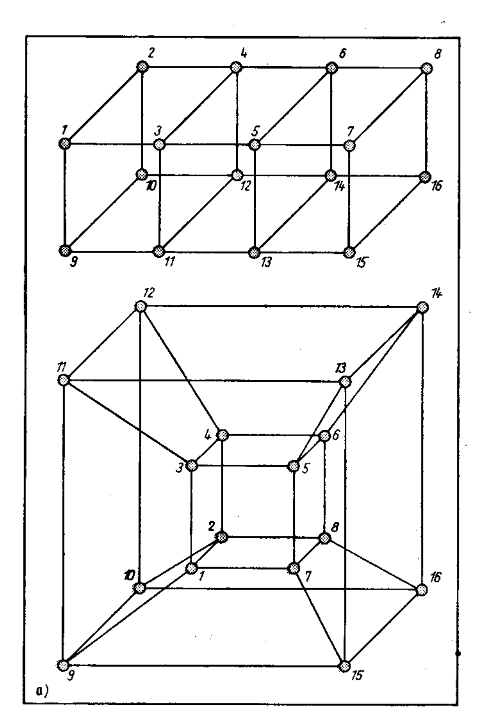


Bild 7 Darstellung eines 2dimensionalen Gitters mittels eines 4dimensionalen Hypercubes.

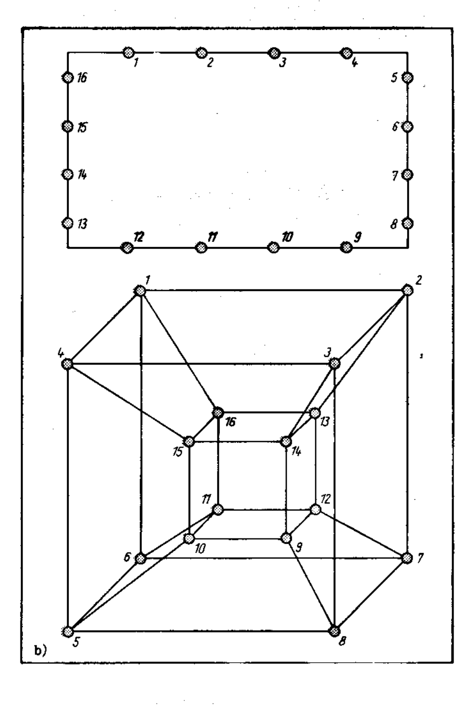


Bild 8 Darstellung eines Ringes mittels Hypercube.

sprache für Rechner mit parallelen Prozessoren darstellt. Die Entwicklung von Algorithmen für Multiprozessorsysteme ist im internationalen Maßstab hinter der Entwicklung der Hardware zurückgeblieben. Um ein Problem auf eine derartige parallele Architektur abzubilden, muß der Programmierer das Problem in Segmente unterteilen, die parallel ausgeführt werden sollen und festlegen, wie die Synchronisation und die Kommunikation der Prozessoren erfolgt. Vom Autor wurde in Kooperation mit der Akademie der Wissenschaften Kiew eine Vielzahl von leistungsfähigen Algorithmen für Multiprozessorsysteme entwickelt /14/.

Ausblick

Die Mikroprozessortechnik ermöglicht den Bau vielfältiger Rechner unterschiedlicher Leistung. Neben den Hochleistungsrechnern in Form der Minisuper-Computer und der Supermini-Computer gibt es eine Vielzahl von Rechnern geringerer Leistung. Für alle gemeinsam ist neben der Hardware entscheidend, daß das jeweilige Problem in geeigneter Weise auf die Hardware abgebildet wird, daß die Software insgesamt der Hardware entspricht. In den nächsten Jahren sind weitere spürbare Leistungserhöhungen sowohl durch noch leistungsfähigere Mikroprozessoren als auch durch neue Architekturkonzepte zu erwarten. Dabei werden auch die Hochleistungsrechner näher an den Arbeitsplatz gebracht werden und zu einem Teil die heutigen Arbeitsstationen ablösen. Kennzeichnend hierfür ist die Entwicklung von sogenannten Superworkstations. Dies sind Einzelnutzersysteme mit extrem hoher Rechnerleistung und Hochleistungsgrafik.

Literatur

/1/ Neubert, P.: CeBit '88. Mikroprozessortechnik 2 (1988) 9, S. 282
/2/ Dongarra, J.: A Survey of High Performance Computers. Compcon Spring 86, Digest of Papers, IEEE Computer Society Press Washington, 1986, S. 1
/3/ Ohr, St.: New Minis Bring Supercomputers to Earth. Electronic Design 35 (1987) 10, S. 33
/4/ Wallich, P.; Zorpette, G.: Minis and Mainframes. IEEE Spectrum (1987) 1, S. 28
/5/ Gaudiot, J. L.; Dubois, M.; Lee, L.-T.; Tohme, N. G.: The TX16: A Highly Programmable Multi-Microprocessor Architecture. IEEE Micro 6 (1986) 5, S. 18
/6/ Ameling, W.: Das Multiprozessorsystem M^5PS: Einleitung. Angewandte Informatik (1987) 8/9, S. 317
/7/ Hartung, G.; Milde, J.; Ameling, W.: Organisation eines M^5PS-Teilsystems. Angewandte Informatik (1987) 8/9, S. 328
/8/ Krings, L.; a Campo, M.; Ameling, W.: Das M^5PS-Teilsystem: Ein eng gekoppelter Multiprozessor. Angewandte Informatik (1987) 8/9, S. 319
/9/ Seitz, Ch. L.: The Cosmic Cube. Communications of the ACM 28 (1985) 1, S. 22
/10/ Stopp, A.: Hypercube-Architecturen. INFO 88 22.-26. Februar 1988 Dresden, Vortragsband, GI der DDR, Berlin 1988, S. 67
/11/ Welty, L. L.; Patton, P. C.: Hypercube Architectures. AFIPS Conference Proceedings 1986 National Computer Conference, AFIPS Press Reston Virginia 1986, S. 495
/12/ Wiley, P.: A Parallel Architecture Comes of Age at Last. IEEE Spectrum 24 (1987) 6, S. 46
/13/ Heuer, H.: Parallelverarbeitende Rechnersysteme. Mikroprozessortechnik 1 (1987) 3, S. 71
/14/ Heuer, H.: Die Lösung wissenschaftlich-technischer Aufgaben auf Rechnersystemen mit Parallelverarbeitung. Technische Universität Magdeburg 1987, als Manuskript gedruckt

⊠ KONTAKT ☏

Technische Universität Magdeburg, Sektion Informatik, Boleslaw-Bierut-Platz 5, Postfach 124, Magdeburg, 3010; Tel. 59 24 29

TERMINE

KDT-Fachtagung

CAD/CAM-Software und Grafikanwendungen

WER? Fachunterausschuß „Applikation von Minicomputersystemen"
WANN? 30. Mai–1. Juni 1989
WO? Neubrandenburg
WAS? CAD-Software, Datenbanken für CAD, Grafiksoftware (GKS), Software für Produktions- und Laborautomatisierung für 16-Bit-Rechenanlagen des SKR wie SM14–20, SM52–11, K1630, I102F u. a.
WIE? Anmeldung bei KDT Neubrandenburg, Sponholzer Straße 9, Neubrandenburg, 2000

Prof. Dr. Th. Horn

3. Internationale Konferenz CAIP '89

(Computer Analysis of Images and Patterns)

WER? KDT, AdW, Gesellschaft für Informatik und Karl-Marx-Universität Leipzig
WANN? 31. 8. bis 2. 9. 1989
WO? Leipzig
WAS?
- schnelle Signalverarbeitung
- Bildmodellierung
- Algorithmen
- Künstliche Intelligenz für das Bildverständnis

WIE? Teilnahme- bzw. Vortragsmeldungen bitte an Präsidium der KDT, Konferenzsekretariat CAIP '89, Clara-Zetkin-Straße 115-117, Berlin, 1086

Huth

4. Tagung „Elektronik-Technologie" aus Anlaß des 20jährigen Bestehens der Sektion Elektronik der Humboldt-Universität

WER? Sektion Elektronik der Humboldt-Universität zu Berlin
WANN? 5. bis 11. November 1990
WO? Berlin
WAS? Technologie mikro- und optoelektronischer Systemelemente, Prozeßautomatisierung, Optische Nachrichtentechnik, Technische Informatik u. a.
WIE? Anfragen an Humboldt-Universität zu Berlin, Sektion Elektronik, Vorbereitungskomitee 4. TET, Invalidenstraße 110, Berlin, 1040; Tel. 2 80 35 94

Dr. Hochmuth

Dynamischer Dual-Port-RAM koppelt Mikrorechner

Elke Thomä, Bernd Däne,
Dr. Wolfgang Fengler
Technische Hochschule Ilmenau

In Multiprozessorsystemen sind häufig Rechnerbaugruppen miteinander zu verbinden, wobei unterschiedliche Prozessortypen und Bussysteme vorliegen können. Je nach Einsatzzweck werden unterschiedliche Arten der Kopplung realisiert. Danach werden die Systeme in zwei Klassen eingeteilt: Multiprozessorsysteme mit loser und mit enger Kopplung /1/.
Lose gekoppelte Systeme sind solche, bei denen kein gemeinsamer Speicherbereich vorhanden ist. Eine Möglichkeit der losen Kopplung zwischen zwei Systemen ist eine serielle Schnittstelle, wobei die Geschwindigkeit der Datenübertragung nicht sehr hoch ist. Eine zweite Möglichkeit ist die Kopplung über einen bidirektionalen FIFO-Speicher. Diese Variante ist bei Systemen vorteilhaft, die mit unterschiedlicher Geschwindigkeit arbeiten und bei denen der Datenaustausch nicht kontinuierlich, sondern bündelweise abläuft (burst).
Eng gekoppelte Systeme werden in solche mit gemeinsamem Bus (an den unter anderem der gemeinsame Speicher angeschlossen ist) und solche ohne gemeinsamem Bus (aber mit gemeinsamem Speicher) unterteilt. Der Nachteil der Multiprozessorsysteme mit gemeinsamem Bus liegt darin, daß der Bus zu einem Zeitpunkt immer nur von einem System genutzt werden kann, was zu beträchtlichen Wartezeiten führen kann. Wenn Systeme über einen gemeinsamen Dual-Port-RAM (DPR) gekoppelt sind, über den die Kommunikation zwischen den Systemen abläuft, verfügt jedes System über einen eigenen Bus und eigene Lokalspeicher. Wartezeiten können nur dann auftreten, wenn zur gleichen Zeit beide Systeme den Speicher ansprechen.

Anforderungen an Dual-Port-RAMs

Je nach Einsatzfall werden DPRs mit unterschiedlicher Leistungsfähigkeit verwendet. Dementsprechend ist der für die Realisierung zu treibende Aufwand sehr unterschiedlich.

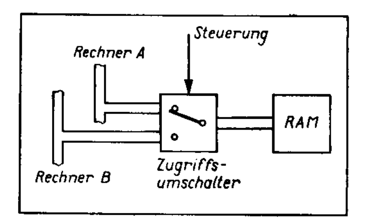
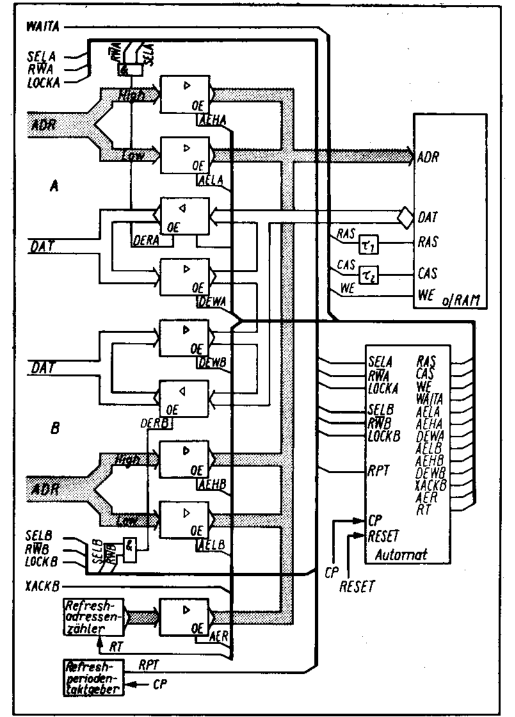


Bild 1 DPR mit exklusiver Zugriffssteuerung

Eine einfache Ausführungsform ist in Bild 1 dargestellt. Der RAM-Block wird mit Hilfe eines elektronischen Umschalters entweder dem einen oder dem anderen Bussystem zugeschaltet. Danach besitzt dieses Bussystem temporär das exklusive Zugriffsrecht für den RAM. Die Steuerung des Umschalters erfolgt gewöhnlich über Anforderungssignale, die unter Kontrolle von Softwarekomponenten der beteiligten Rechner mit Hilfe spezieller Baugruppen (z. B. E/A-Tore) generiert werden. Jeder Rechner muß vor einem Zugriff oder einer Zugriffsfolge das Zugriffsrecht anfordern und bestätigen lassen. Der dazu benötigte Zeitaufwand ist um so störender, je größer die durchschnittliche Umschaltrate des Zugriffsumschalters wird. Daher werden mit der gezeigten Lösung nur in Fällen mit vielen Zugriffen pro Umschaltzyklus und/oder relativ geringem Gesamtdurchsatz gute Ergebnisse erzielt.
Die nachfolgend beschriebene Konzeption für einen DPR geht von einem hohen Gesamtdurchsatz bei abwechselnder Übertragung meist nur weniger Bytes aus (typischer Fall bei wechselseitiger Prozeßkommunikation) und benutzt daher das Prinzip des quasi gleichzeitigen (gegenseitig transparenten) Zugriffs. Dieses Prinzip erlaubt jedem Rechner zu jedem beliebigen Zeitpunkt die Durchführung eines den DPR ansprechenden Speicherzyklus. Ist der DPR verfügbar, so wird er dem anfordernden Rechner sofort zugeteilt. Läuft dagegen bereits ein Zugriff des anderen Rechners, so wird der Speicherlese- oder -schreibzyklus des anfordernden Rechners durch entsprechende Steuersignale vorübergehend angehalten, bis der DPR zur Verfügung steht. Die Abweisung eines Zugriffs ist prinzipiell nicht möglich. Die Zugriffe erfolgen damit aus der Sicht der Programme ungehindert und scheinbar gleichzeitig. Der Aufwand für die Beantragung und Abfrage der Zugriffserlaubnis entfällt. Demgegenüber tritt eine Verlängerung der durchschnittlichen Speicherzugriffszeit ein, die besonders dann spürbar wird, wenn beide Rechner ständig sehr häufig zugreifen. Auch ist der Aufwand für die Realisierung der Zugriffssteuerung vergleichsweise hoch.
Aufgrund der mit dynamischen RAM-Schaltkreisen erreichbaren hohen Speicherkapazitäten ist das nachfolgend beschriebene Konzept auf die Verwendung dieser Schaltkreise ausgerichtet, obwohl die Besonderheiten der zu generierenden Speicherzyklen und die Notwendigkeit der periodischen Regenerierung des Speicherinhalts zu zusätzlichen Problemen führen.

Bild 2 Blockschaltbild des DPR mit quasi gleichzeitigem Zugriff



Struktur des Dual-Port-RAMs

Das Blockschaltbild in Bild 2 zeigt die Struktur des dynamischen Dual-Port-RAMs, der die Busse der zwei Rechnereinheiten A und B miteinander koppelt. Der DPR gliedert sich in die drei Teile *Speicherblock aus dynamischen RAMs (dRAM)*, *Steuerlogik* und *Treiber*. Die gewählten Signalnamen beziehen sich nicht auf ein bestimmtes Buskonzept und sind sinngemäß anzuwenden. Beispielsweise ist **SELECT** (**SELA**, **SELB**; s. Tafel 1) das Anforderungssignal für einen Speicherzugriff (im allgemeinen ein Ausgangssignal eines Adreßdekoders). Die Signale sind alle *high-aktiv* dargestellt. Für die beiden auf den DPR asynchron zugreifenden Rechner soll gelten, daß die Speicherzyklen durch spezielle Signale verlängert werden können. Im vorliegenden Beispiel sind dafür zwei ver-

Tafel 1 Erläuterung der Signalnamen

Signalname	Kurz-bezeichnung	Bedeutung
SELECT A SELECT B	SELA SELB	Speicheranwahl-signale
READ/$\overline{\text{WRITE}}$ A READ/$\overline{\text{WRITE}}$ B	R$\overline{\text{W}}$A R$\overline{\text{W}}$B	Unterscheidung von Schreib- und Lesezugriffen
LOCK A LOCK B	LOCKA LOCKB	Anforderung unteilbarer Zugriffsfolgen
WAIT A	WAITA	Verlängerung des Buszyklus
EXECUTION ACKNOWLEDGE B	XACKB	Beendigung des Buszyklus
ROW ADDRESS SELECT COLUMN ADDRESS SELECT WRITE ENABLE	RAS CAS WE	Steuersignale für die dRAMs
CLOCK PULSE	CP	Automatentakt
RESET	RESET	Herstellung des Initialzustandes
DATA ENABLE READ A DATA ENABLE READ B DATA ENABLE WRITE A DATA ENABLE WRITE B ADDRESS ENABLE HIGH A ADDRESS ENABLE LOW A ADDRESS ENABLE HIGH B ADDRESS ENABLE LOW B ADDRESS ENABLE REFRESH	DERA DERB DEWA DEWB AEHA AELA AEHB AELB AER	Freigabesignale für Treiber
REFRESHTAKT REFRESHPERIODEN-TAKT	RT RPT	Taktsignal für Refreshadressenzähler Refreshperiodentakt vom Refreshperiodentaktgeber
M1 A REFRESH A	M1A RFSHA	Steuersignale eines U 880 (Bild 5)

Anmerkung: Unabhängig von einer denkbaren schaltungstechnischen Realisierung werden hier alle Signale als high-aktiv betrachtet. Die Endbuchstaben **A** und **B** von Signalnamen weisen auf die Zugehörigkeit der Signale zu den Rechnern A oder B hin.

schiedene Möglichkeiten dargestellt. Für den Rechner A wird ein Wartesignal (**WAITA**) vom Zyklusanfang bis zur Zugriffsbewilligung und für den Rechner B ein Bestätigungssignal (**XACKB**) generiert, das den Abschluß des Speicherzyklus' bewirken soll.
Die Steuerung des wechselseitigen Zugriffs und die Generierung der benötigten Steuersignale wird durch einen getakteten Zugriffsautomaten übernommen. Der Automatentakt kann zu den Takten beider Rechner asynchron sein. Seine Periodendauer darf nicht kleiner sein als das Maximum aus CAS-Zugriffszeit, halber RAS-Zugriffszeit und Erholzeit der verwendeten dRAM-Bausteine. Zur Absicherung der Adreßvorhaltezeiten für die Zeilen- und die Spaltenadresse werden die beiden Signale RAS und CAS durch entsprechende Schaltungen um die (gegenüber der Automatentaktperiode kurzen) Zeiten τ_1 und τ_2 verzögert. Das Regenerieren (Refresh) des dRAM-Blocks realisiert der Automat in Zusammenarbeit mit einem Refreshadressenzähler und einem Refreshperiodentaktgeber.
Beide Rechner haben die Möglichkeit, durch ein **LOCK**-Signal (**LOCKA, LOCKB**) vorübergehend alle Zugriffe des jeweils anderen Rechners zu unterbinden (z. B. für nichtunterbrechbare Test- und Setz-Befehle). Refreshzugriffe sind davon nicht betroffen. Die Treiberschaltkreise realisieren den Anschluß an die Adreß- und Datenbusse und an den Refreshadressenzähler. Die Ausgänge der Treiber werden mit den Signalen an ihren OUT-Enable (OE)-Eingängen (**AEHA, AELA, DERA** usw., vgl. Tafel 1) aktiviert. Sind diese Signale inaktiv, so nehmen die Ausgänge einen hochohmigen Zustand ein. Besonderheiten von Systembussen bezüglich unterschiedlicher Datenbreite und evtuellen Multiplexbetriebs wurden nicht berücksichtigt.

Zugriffssteuerung für die Rechner A und B

In Bild 3 ist ein Petrinetz /2/ dargestellt, welches das Verhalten des zugriffssteuernden Automaten beschreibt. Im Netz sind bei den Plätzen Gleichungen für die Ausgangsvariablen angegeben, welche bei Markierung des betreffenden Platzes gültig werden. Nicht dargestellt sind dabei die jeweils mit *Null* belegten Ausgangsvariablen. Bei mehreren gleichzeitig markierten Plätzen soll die *ODER*-Verknüpfung der diesen Plätzen zugeordneten Ausgangsbelegungen gelten.
Ein Rechner, der den Zugriff durch sein Signal SELA bzw. SELB anfordert, erhält die Zugriffsbewilligung, wenn der DPR für ihn verfügbar ist. Für den Spezialfall einer (im Rahmen des Taktrasters) gleichzeitigen Speicheranforderung wird dem Rechner A die höhere Priorität eingeräumt. Jeder begonnene Speicherzyklus wird bis zur erfolgreichen Ausführung des Zugriffs verlängert. Für den Rechner A wird zu diesem Zweck innerhalb des unteren Teilnetzes ein Wait-Signal generiert, das mit der Zugriffsanforderung von Rechner A aktiviert und erst bei erfolgtem Zugriff zum RAM abgeschaltet wird. Bei Buskonzepten, wie sie beispielsweise die Prozessoren U880 und U8000 benutzen, läßt sich so die notwendige Verlängerung des Speicherzyklus' erreichen. Andere Prozessoren, z. B. der 8086 (K1810 WM86), verharren in jedem Zyklus so lange, bis ein Bestätigungssignal die Ausführung der angeforderten Operation anzeigt. In dem für den Zugriff des Rechners B zuständigen Teil des Netzes

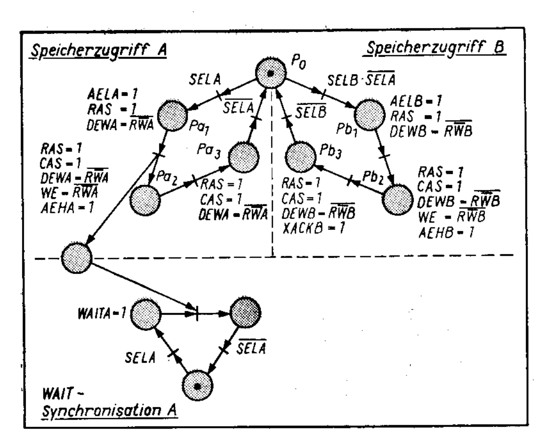


Bild 3 Petrinetz für die Zugriffssteuerung des DPR

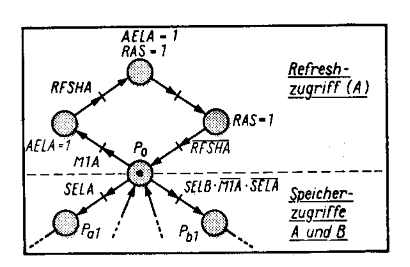


Bild 5 Teilnetz zur Beschreibung der modifizierten Refreshfunktion bei Verwendung eines U880 im Rechner A

wird die Bildung dieses Signals, hier als **XACKB** bezeichnet, gezeigt.

Regenerieren (Refresh) des dynamischen DPR

Die Generierung der Refreshzyklen für den DPR realisiert der Zugriffsautomat in Zusammenarbeit mit dem Refreshadressenzähler und dem Refreshperiodentaktgeber (s. Bild 2). Das in Bild 4 dargestellte Teilnetz erweitert das Petrinetz aus Bild 3 um eine komfortable Refreshfunktion.
Anmerkung: Bei der Vereinigung beider Netze sind gleichbezeichnete Plätze als identisch zu betrachten.
Bei Nichtvorhandensein von Zugriffsanforderungen der Rechner A und B werden fortlaufend Refreshzyklen (RAS-only-Refresh) ausgeführt, indem die Signale **AER** und **RAS** generiert werden. Die benötigte Zeilenadresse wird dabei vom Refreshadressenzähler geliefert, welcher durch das Signal RT inkrementiert wird. Ein Zugriff eines der beiden Rechner unterbricht diesen Vorgang. Ist das häufig der Fall, so ist die Anzahl der ausgeführten Refreshzyklen nicht mehr ausreichend. Deshalb muß die Häufigkeit dieser Refreshzyklen überwacht und bei Bedarf das Erzwingen zusätzlicher Refreshzyklen ermöglicht werden. Das wird durch den oberen Teil des Netzes in Bild 4 realisiert.
Der Refreshperiodentaktgeber aktiviert in regelmäßigen Abständen das Signal **RPT**. Die Periodendauer ist dabei so gewählt, daß sie der Hälfte der Zeit entspricht, die für das Regenerieren einer bestimmten (mit n bezeichneten) Anzahl aufeinanderfolgender Zeilen im Speicherblock maximal zulässig ist. Der Maximalwert für n ist die Gesamtzahl der zu regenerierenden Speicherzeilen. In diesem Fall entsprechen dem Zeitraum, für den das Regenerieren aller Zeilen des Speichers gefordert wird, zwei Impulse von **RPT**. Wird n kleiner gewählt, so muß die Impulsfrequenz

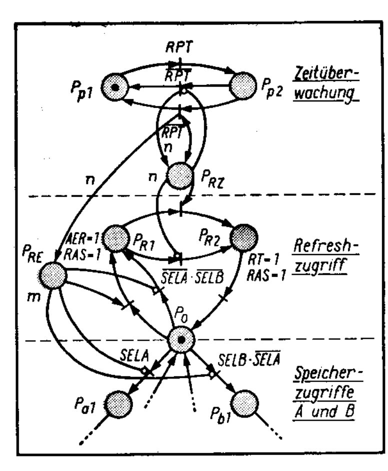


Bild 4 Teilnetz zur Beschreibung der Refresh-Funktion

entsprechend vergrößert werden. Wird der Minimalwert von n = 1 gewählt, so entspricht die Taktperiode von **RPT** der Hälfte der Zeit, innerhalb der bei regelmäßigem fortlaufendem Regenerieren der Speicherzeilen wenigstens eine Zeile regeneriert werden muß. Die Gesamtheit der beschriebenen Abläufe ermöglicht das Regenerieren des DPR unter weitgehender Ausnutzung anforderungsfreier Zeiten, wobei die Einhaltung der Refreshzeitbedingungen überwacht und notfalls durch zusätzlich eingeschobene Refreshzugriffe erzwungen wird. Diese können sowohl in größeren Zeitabständen für eine große Anzahl Speicherzeilen als auch in kurzen Abständen für jeweils nur eine oder wenige Speicherzeilen durchgeführt werden. Beide Varianten weisen Vor- und Nachteile bezüglich des Zugriffszeitverhaltens für die Rechner A und B auf.
Anmerkung: Die an beiden Enden mit Pfeilen versehene Kante ist eine Testkante. Sie macht das Schalten der Transition vom Markierungszustand des Platzes abhängig, ohne diesen jedoch zu verändern (siehe /2/).
Einen Spezialfall stellt das Petrinetz in Bild 5 dar. Die (hier im Rechner A eingesetzte) CPU U880 kann das Regenerieren von dynamischen RAMs selbst durchführen /3/. In jedem Befehlsholezyklus (M1-Zyklus) wird nach dem Holen des Befehls ein Refreshzyklus generiert, der für das Regenerieren des DPR genutzt werden kann. In diesem Refreshzyklus wird jedoch das Wait-Signal ignoriert, so daß er sich nicht verlängern läßt. Um zu sichern, daß jeder von der CPU ausgesendete Regenerierzyklus einen **RAS**-Only-Refreshzyklus im DPR bewirkt, muß dieser Zyklus dem Zugriffsautomaten rechtzeitig durch ein geeignetes Signal vorangekündigt werden, woraufhin dieser die Bewilligung von Zugriffen des anderen Rechners einstellt.
Für die Vorankündigung kann das **M1**-Signal der CPU verwendet werden. Bei Aktivwerden von **M1** wird jeder neue RAM-Zugriff gesperrt und somit die Durchführung des **RAS**-Only-Refreshzyklus sofort nach Aktivwerden des Signals **RFSH**, das den Refreshzyklus der CPU kennzeichnet, gesichert. Dabei wird vorausgesetzt, daß der Speicherzugriff im Befehlszyklus den DPR nicht berührt, das heißt, der DPR darf keine Programmkomponenten des Rechners A enthalten. *Damit schließen sich* **SELA** *und* **M1A** *gegenseitig aus.*

LOCK-Funktion

Um die Möglichkeit der vorübergehenden exklusiven Verfügbarkeit des DPR für einen der beiden Rechner zu schaffen, kann eine

LOCK-Funktion eingefügt werden. Diese wird durch das Petrinetz in Bild 6 beschrieben.

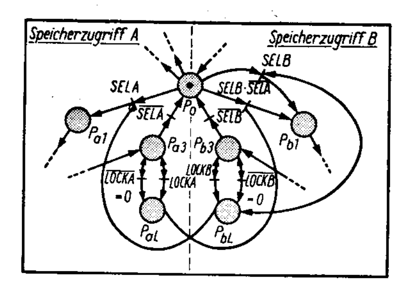


Bild 6 Teilnetz zur Beschreibung der LOCK-Funktion

Anmerkung: Die mit „=0" bezeichneten Kanten sind Rücksetzkanten. Sie entleeren beim Schalten der Transition den angeschlossenen Platz, verhindern aber bei bereits leerem Platz das Schalten der Transition nicht. Analog werden mit „=1" Setzkanten bezeichnet /2/.

Integrierte DPR-Controller

Auf dem Weltmarkt existieren integrierte Schaltkreise, welche die komplette Ansteuerung von Dual-Port-RAMs übernehmen können. Ein herausragendes Beispiel ist der zweitorfähige dRAM-Controller 8207 /4/.
Dieser Schaltkreis ermöglicht die Ansteuerung dynamischer RAM-Schaltkreise der Größen 16K*1Bit, 64K*1Bit und 256K*1Bit. Diese können in maximal 4 Bänken bei beliebiger Datenwortbreite angeordnet sein. Der 8207 generiert die Signale **RAS**, **CAS** und **WE** für die dRAMs sowie einige Steuersignale für die externe Logik. Der Adreßmultiplexer für die Umschaltung zwischen Zeilen- und Spaltenadresse ist im Schaltkreis enthalten. Bei Zweitorbetrieb müssen die Adreß- und Datenleitungen der beiden Busse unter Steuerung des 8207 extern umgeschaltet werden. Die Erkennung der Speicheranforderung erfolgt beim 8207 über Eingänge, die wahlweise die Statussignale der Prozessoren 8086, 8088, 80186 und 80286 oder die Speicherkommandos von durch diese Prozessoren gesteuerten Bussen verarbeiten können. Verschiedene Zeitparameter können modifiziert werden, so daß in Abhängigkeit von den Taktfrequenzen, Prozessortypen und Zugriffszeiten das Zeitverhalten optimiert werden kann. Der 8207 enthält einen Refreshadressenzähler und einen Refreshzeitgeber, so daß er das Regenerieren der dRAMs selbständig ausführen kann. Es ist jedoch auch möglich, Refreshs (oder Refresh-Bursts) durch externe Signale auszulösen, wobei der 8207 auf Wunsch die Einhaltung der Refreshzeitbedingungen überwachen kann. Die Ablaufsteuerung für die Speicherzugriffe und Refreshvorgänge erfolgt über einen im 8207 enthaltenen Arbiter, welcher die Zugriffserteilung wahlweise priorisiert oder nichtpriorisiert vornehmen kann.
Weitere Funktionen des 8207 betreffen eine mögliche Zusammenarbeit mit dem Fehlererkennungs- und -korrekturschaltkreis 8206 /4/, der den Aufbau von Speichereinheiten mit automatischer Fehlererkennung und -korrektur (auf der Grundlage zusätzlicher Korrekturbits) ermöglicht.
In Bild 7 ist als Beispiel eine Anwendungsschaltung mit dem 8207 zur Ansteuerung eines DPR von insgesamt 512 kByte Speicherkapazität dargestellt /4/. Die gekoppelten Rechnereinheiten A und B benutzen beide den Mikroprozessor 8086, der im Maximum-Mode zusammen mit dem Systemtaktgeber 8284A und dem Buscontroller 8288 betrieben wird. Für Rechner A ist die Ansteuerung über die Statusleitungen der CPU und für Rechner B die Benutzung der Speicherkommandos des Buscontrollers 8288 demonstriert. Der Refresh des aus 4 Bänken mit je 16 Speicherschaltkreisen der Größe 64K* 1Bit bestehenden dRAM-Blocks wird vom 8207 selbstänig vorgenommen.
Durch den Einsatz von hochintegrierten Zusatzschaltkreisen wie dem 8207 kann die Schaltungstechnik von Baugruppen für Mikrorechner vereinfacht und die Leistungsfähigkeit und Zuverlässigkeit wesentlich erhöht werden.

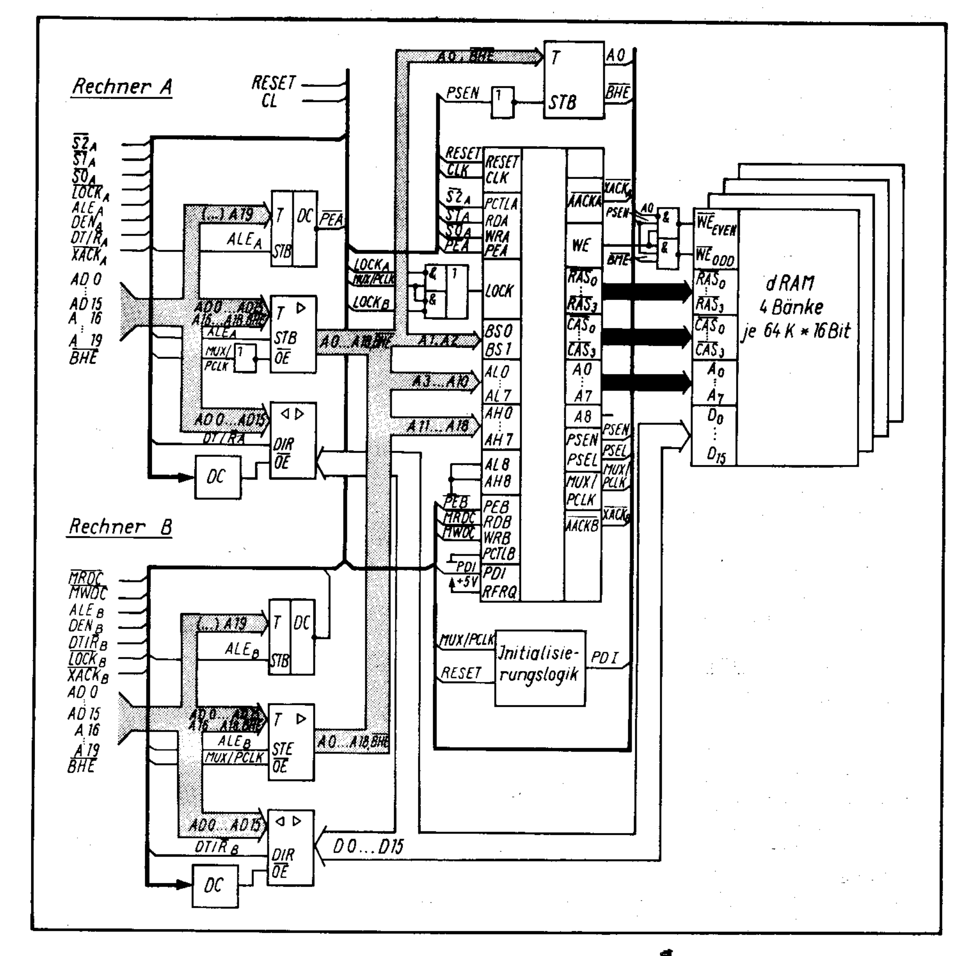


Bild 7 Schaltungsbeispiel für einen DPR mit dem Schaltkreis 8207 /4/

Literatur

/1/ Gandhi, S.: Mikrocontroller koppelt Prozessoren. Elektronik (1987) 12, S. 77

/2/ Fengler, W.: Entwurf und Technik von Mikroprozeßrechnern. Dissertation B. TH Ilmenau, 1988

/3/ Kieser, H.; Meder, M.: Mikroprozessortechnik, Aufbau und Anwendung des Mikroprozessorsystems U 880. Berlin: VEB Verlag Technik 1985

/4/ Intel Component Data Catalog 1983. Intel Corporation, Santa Clara 1983

✉ KONTAKT ☎

Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Computertechnik PSF 327, Ilmenau 6300; Tel. 7 48 43 (E. Thomä)



Nutzereigene Laufzeitfehlerbehandlung bei mathematischer Software

Dr. Klaus-Dieter Weber
Technische Hochschule Ilmenau

Der Entwickler mathematischer Software hat i. allg. in den höheren Programmiersprachen a priori keine Möglichkeit, mit vernünftigem Aufwand die möglicherweise während der Laufzeit, z. B. in einem Bibliotheksmodul, auftretende Arithmetikfehler auszutesten. Bei unerwünschten, d. h. vermeidbaren Fehlern können somit entsprechende Maßnahmen nicht ergriffen werden. Infolgedessen steht der Nutzer mathematischer Software häufig recht hilflos einer Laufzeitfehlermeldung aus einem von ihm nicht implementierten Bibliotheksmodul gegenüber. Wünschenswert wäre deshalb für den Softwareentwickler die Möglichkeit einer eigenen Laufzeitfehlerbehandlung (anstatt durch den Compiler) einschließlich eigener Fehlermeldungen und Fehlerlokalisation, damit eine an der Nutzeroberfläche deutbare Fehlermeldung abgesetzt werden kann. In diesem Artikel soll deshalb ein erweiterungs- und verallgemeinerungsfähiges Konzept zur Realisierung einer nutzereigenen Laufzeitfehlerbehandlung in höheren Programmiersprachen auf PCs vorgestellt und seine Realisierung am Beispiel der Arithmetikfehlerbehandlung unter Turbo-Pascal gezeigt werden.

Anforderungen an nutzereigene Fehlerbehandlung

In allen höheren Programmiersprachen signalisieren die Compiler während der Laufzeit das infolge einer Arithmetikoperation aufgetretene Verlassen des darstellbaren Zahlenbereiches als Zahlenüberlauf:

real-overflow und zum Teil auch *integer overflow* (01)

(Ein Unterschreiten des zulässigen Zahlenbereiches liefert im allgemeinen eine Null als Ergebnis und keine Fehlermeldung). Ebenso wird der Versuch, durch Null zu dividieren, angezeigt:

zero-division (02)

Eine Überprüfung des Argumentes der Standardfunktionen zur Berechnung der Quadratwurzel und des (natürlichen) Logarithmus findet ebenfalls statt und führt bei nicht zulässigem Argument zu der Fehlermeldung:

SQRT-argument-error bzw. *LN-argument-error* (03)

Laufzeitfehler führen entweder zum sofortigen Programmabbruch oder erzeugen eine im allgemeinen gerade aussagekräftige (Folge-) Fehlerliste.

Das *vorausschauende* Austesten von unerwünschten Arithmetiklaufzeitfehlern bei Rechenoperationen ist vielfach – wenn überhaupt – nur mit unvertretbarem Aufwand möglich. Insbesondere ist der Überlauffehler (01) von der internen Zahlendarstellung und somit implementationsabhängig, das heißt u. a., daß Numerikprogramme, die sich an der Maschinengenauigkeit orientieren, häufig noch nicht einmal innerhalb einer Sprache von einem Rechner auf einen anderen problemlos portierbar sind.

Zahlreiche Einzelüberprüfungen bei Rechenoperationen würden überflüssig, falls sich die Fehlerbehandlung durch den Compiler an kritischen Programmstellen ausschalten (und wieder einschalten) ließe und somit eine *nachträgliche* Fehlerbehandlung möglich wäre.

Eine damit durch den Softwareentwickler mögliche Fehlerbehandlung bietet zwei grundsätzliche Vorteile:

- Einfache Reparierbarkeit von unerwünschten Ergebnissen aus Operationen, die zu Laufzeitfehlern führen würden
- Absetzen beliebiger Laufzeitfehlermeldungen zu beliebiger Laufzeit mit Fehlerlokalisation an (fast) jeder Stelle.

Die Anforderungen, die an eine solche „Spracherweiterung" gestellt werden, sind im folgenden „Pflichtenheft" aufgelistet:

Die Erweiterung soll

1. den Compiler bis auf das notwendige „Patchen" von Sprungleisten unverändert lassen
2. fakultativ und nachladbar sein
3. so implementiert sein, daß ihre Prozeduren vor Fehlbedienung geschützt sind (protected procedures) und die Prozeduren auf einer gemeinsamen Datenkapsel (protected area) arbeiten
4. ein- und ausschaltbar sein (ohne daß eine Startinitialisierung nötig ist): ERROR_ON, ERROR_OFF
5. die Fortsetzung des Programms nach einer einen Laufzeitfehler erzeugenden Operation mit einem genau festgelegten Ergebnis bewirken (bei eingeschalteter Erweiterung)
6. das Abfragen nach aufgetretenen Fehler(nummer)n seit Aktivierung der Erweiterung ermöglichen: RUN_RESULT
7. das Absetzen beliebiger (eigener) Fehlermeldungen von beliebiger Programmstelle ermöglichen: RUNTIME_ERROR
8. das Setzen des Fehlerzeigers an (fast) jede beliebige Stelle im Programm ermöglichen, aber insbesondere das Zeigen auf den Aufruf einer Prozedur, in der ein Laufzeitfehler infolge ungünstiger Übergabeparameter auftritt (ähnlich der Original-Fehleranzeige des Compilers für die Fehlergruppe 03).

Implementation einer Arithmetikfehlerbehandlung unter TURBO-PASCAL 3.00[1]

Für die Implementation einer nutzereigenen Fehlerbehandlung werden die folgenden Informationen benötigt:

I1. Adresse(n) der Original-Fehlerbehandlungsroutine(n)

I2. Übergabeparameter für die Original-Fehlerbehandlungsroutine (in den Registern und/oder auf dem Stack).

Speziell für eine nutzereigene Arithmetikfehlerbehandlung muß die Frage noch beantwortet werden:

F1. Wie wird das Ergebnis einer arithmetischen Operation zur Weiterverarbeitung übergeben (register- oder stackorientiert)?

Die beiden Informationen I1 und I2 erhält man sehr einfach durch Debugging eines

Tafel 1 Dokumentation ERROR.BIB

Behandlung von arithmetischen Laufzeitfehlern Var. 1.7c (C) 11.01.88 by Wb	
1.	procedure *ERROR_STOP* stoppt die Abarbeitung eines Programms und positioniert den Fehlerzeiger hinter den Aufruf dieses Stopps.
2.	procedure *RUNTIME_ERROR* (No:byte; var remark; depth:byte) stoppt ebenfalls die Abarbeitung unter Angabe einer vom Nutzer angebbaren Fehlernummer **no** und einer verbalen Fehlerbeschreibung, die über den ungetypten Variablenparameter remark übergeben wird. Der Fehlerzeiger wird hinter den Aufruf der den Fehler verursachenden procedure (function) gesetzt. Hierbei ist mit dem Parameter depth (depth≧0) der ‚prozedurale Abstand' (Eindringtiefe) zwischen Fehlerstelle und angezeigter procedure (function) einstellbar. *Anmerkung:* **No** ist (sinnvollerweise) als hexadezimale Zahl anzugeben. Beim Aufruf der procedure muß dem aktuellen Parameter für remark eine Stringvariable (die auch leer sein kann) zugewiesen werden.
3.a	function *RUN_RESULT:* integer; wirkt wie eine vordefinierte (nur abfragbare) Variable und kann die folgenden Werte annehmen: (falls ERROR_ON siehe 3.b) RUN_RESULT = 0 → kein Arithmetikfehler (voreingestellt) RUN_RESULT > 0 → Arithmetikfehler, dann enthält Lo (RUN_RESULT) den ersten und Hi (RUN_RESULT) den letzten zwischen ERROR_ON und Abfrage von RUN_RESULT aufgetretenen Arithmetikfehler. Es werden die Original-Fehlernummern des Compilers für die folgenden Fehlerarten übergeben: Gleitkommaüberlauf Division durch Null SQRT-argument <0 LN-argument ≦0 außerhalb des Ganzzahlbereiches
3.b	procedure *ERROR_ON* schaltet die nutzereigene Fehlerbehandlung für die unter 3.a angegebenen Arithmetikfehler ein: Der Programmierer hat die Möglichkeit, die Fehlerbehandlung und/oder die Fehleranzeige selbst zu steuern (mittels RUNTIME_ERROR). *Anmerkung:* Das Ergebnis einer arithmetischen Operation, die zu einem RUN_RESULT > 0 führt, wird immer Null gesetzt! RUN_RESULT wird bei jedem Aufruf von ERROR_ON gelöscht.
3.c	procedure *ERROR_OFF* stellt den Originalzustand für die Fehlerbehandlung wieder her. RUN_RESULT bleibt (für spätere Abfragen) erhalten. *Anmerkung:* ERROR_ON und ERROR_OFF können in beliebiger Reihenfolge aufeinanderfolgen. Vor Verlassen eines Nutzerprogramms ist es sinnvoll, ERROR_OFF auszuführen, falls nicht weiter mit ERROR.BIB gearbeitet wird.

[1] Der Wunsch nach einer eigenen Laufzeitfehlerbehandlung für Turbo-Pascal taucht bereits für die Version 2.00 unter CP/M in /1/ auf. Unter Turbo-Pascal ist zwar eine eigene Fehlerbehandlung (User break, I/0 error, Runtime error) durch Zuweisen der Adresse des eigenen „Error-handlers" an die (vordefinierte) Integervariable „ErrorPtr" möglich, führt aber anschließend zum Programmabbruch, und eine Fortsetzung des Programms setzt eine gründliche Analyse der Turbo-Runtime-Library voraus, da eine Verzweigung in die angesprochene benutzereigene Fehlerbehandlung erst im Laufe der Turbo-Fehlerroutine stattfindet und somit die Übergabe-Parameter (Register und Laufzeitstapel) bereits stark verändert sind. Unter diesen Voraussetzungen ist eine auf Arithmetikfehler beschränkte nutzereigene Fehlerbehandlung für Turbo-Pascal 3.00 unter MS/PC-DOS in /2/ beschrieben. Wegen seiner Allgemeingültigkeit soll hier aber ein anderer Weg beschritten werden.

```
{ ========================================================== }
{        >>>>>>>>>>>>>>>>> ERROR.BIB <<<<<<<<<<<<<<<<<        }
{ ---------------------------------------------------------- }
{        Handling of the run-time errors on TURBO 3.00        }
{        Var. 1.7al                 (C) 11.01.88 by Wb        }
{ ---------------------------------------------------------- }
{        CP/M-80, Z80                           (192 bytes)   }
{ ---------------------------------------------------------- }

function RUN_RESULT: integer;
const result: integer = 0;
begin
  inline(              { so short as possible }
  $2A/result/          { LD HL,result          }
  $C9);                { RET     ; must be!    }
end;

procedure error_handler;
begin
  inline(
  { ---------------------------------------------------------- }
  $C9/                 { RET     | protected procedure!         }
  { ---------------------------------------------------------- }
  $7A/                 { LD A,D  | get error-kind               }
  $B7/                 { OR A    }
  $FE/$02/             { CP 02   | compare with error-kind      }
  $20/$09/             { JRNZ *9 }
  { ---------------------------------------------------------- }
  $7B/                 { LD A,E  | get error-no                 }
  $FE/$92/             { CP 92   | arithmetic run-time errors:  }
  $28/$08/             { JRZ *8  |    >> $01 .. $04, $92 <<     }
  $DE/$05/             { SBC 05  |         ( only )             }
  $38/$04/             { JRC *4  }
  $CD/$7A/$03/         { CALL the patched procedure and .. return }
  $C9/                 { RET     |  to the original error-handler }
  { ---------------------------------------------------------- }
  $E1/                 { POP HL  | remove 'ret' of this procedure }
  $D1/                 { POP DE  | remove error-info from stack   }
  { ---------------------------------------------------------- }
  $7B/                 { LD A,E  }
  $21/run_result+4/    { LD HL,Addr(hi(result))                 }
  $77/                 { LD (HL),A | hi(run_result):=last error  }
  $2B/                 { DEC HL    }
  $7E/                 { LD A,(HL) }
  $B7/                 { OR A      }
  $20/$01/             { JRNZ *1   }
  $73/                 { LD (HL),E | lo(run_result):=first error }
  { ---------------------------------------------------------- }
  $AF/                 { XOR A     | because Z80-CPU is working }
  $67/                 { LD H,A    | on registers only:         }
  $6F/                 { LD L,A    | Clear all registers        }
  $57/                 { LD D,A    }
  $5F/                 { LD E,A    }
  $47/                 { LD B,A    }
  $4F/                 { LD C,A    }
  $DD/$E9)             { JP (IX)   | jump behind error-producer }
end;

procedure ERROR_ON;
var patch: integer absolute $202E;
begin
  patch:=succ(Addr(error_handler));
  { ---------------------------------------------------------- }
  inline(
  $AF/                  { XOR A        }
  $21/run_result+3/     { LD HL,Addr(result) }
  $77/                  { LD (HL),A   } clear the run_result }
  $23/                  { INC HL      }
  $77)                  { LD (HL),A   }
end;

procedure ERROR_OFF;
var patch: integer absolute $202E;
begin
  patch:=$037A;         { restore the original address }
end;

procedure RUNTIME_ERROR(no: byte; var remark; depth: byte);
var message: string[60] absolute remark;
    l: byte absolute remark;
var handler_begin: integer absolute $2027;
begin
  error_off;
  { ---------------------------------------------------------- }
  write(#$0D#$0A,'user-ordered:  ',message);
  { ---------------------------------------------------------- }
  inline(
  $3A/depth/            { LD A,depth| get depth          }
  $47/                  { LD B,A    }
  $04/                  { INC B     }
  $E1/                  { POP HL    | remove 'return'    }
  $10/$FD/              { DJNZ *FD  | depth times        }
  { ---------------------------------------------------------- }
  $23/                  { INC HL    | put PC behind      }
  $23/                  { INC HL    | error-'producer'   }
  $23/                  { INC HL    }
  $23/                  { INC HL    }
  $E5/                  { PUSH HL   | restore PC         }
  { ---------------------------------------------------------- }
  $3A/no/               { LD A,no   | get error-number   }
  $C3/handler_begin)    { jump to TURBOs error-handler   }
end;

procedure ERROR_STOP;
const message: string[10] = 'error-stop';
begin
  Runtime_error($A0,message,1);
end;
{ ========================================================== }
```

Bild 1

```
program test;
{     >> Test-Beispiel fuer die Anwendung der ERROR.BIB <<      }
{ ========================================================== }

function sign(x: real): integer;    { als Zugabe }
{ kurze und schnelle SIGNum-function fuer TURBO-Pascal          }
{                    nur fuer 8 bit-Technik (Z80-CPU)           }
{ ---------------------------------------------------------- }
begin
  inline(
$3A/x/$21/$00/$00/$B7/$C8/$3A/x+5/$CB/$7F/$23/$C8/$2B/$2B/$C9)
end;

function sinh(x: real): real;
{ rechnerunabhaengige Berechnung des Hyperbel-Sinus:            }
{             sinh(x):=(exp(x)-exp(-x))/2                       }
{ mit Fehlermeldung und -lokalisation auf den aktuellen Aufruf  ]
{ ---------------------------------------------------------- }
const message: string[32] = 'argument-error on hyperbel-sinus';
var a: real;
begin
  ERROR_ON;
  a:=exp(x);
  if RUN_RESULT>0 then RUNTIME_ERROR($A0+lo(run_result),message,1);
  sinh:=0.5*(a-1.0/a);
  ERROR_OFF
end;

function tanh(x: real): real;
{ rechnerunabhaengige Berechnung des Hyperbel-Tangens:          }
{            tanh(x):=(exp(x)-exp(-x))/(exp(x)+exp(-x))         }
{ ---------------------------------------------------------- }
var a: real;
begin
  ERROR_ON;
  a:=exp(x+x);
  if RUN_RESULT>0 then tanh:=sign(x) else tanh:=(a-1.0)/(a+1.0);
  ERROR_OFF
end;

{ und so weiter ... }

var z: real;
begin
  ClrScr;
  writeln('kurzes TEST-Programm zur Demonstration von EROR.BIB');
  writeln('--------------------------------------------------');
  writeln;
  repeat
    write(' ':6,'x = '); readln(z);
    writeln('tanh(x) = ',tanh(z));
    writeln('sinh(x) = ',sinh(z));
    writeln;
  until keypressed;
end.
```

Bild 2

eines Programmstückes mit dem Aufruf z. B. der Wurzelfunktion: a:=SQRT(−1.0); denn zu Beginn dieser Standardfunktion steht mit Sicherheit (in Maschinencode): IF 'argument' < 0 THEN GOTO/CALL error_handler; Am Ende dieser (i. a. sehr kurzen) Standardfunktion findet man dann die Antwort auf die Frage F1.

Diese Vorgehensweise besitzt den Vorteil, daß sie auf jede CPU und jeden Compiler angewandt werden kann und auf beliebige Laufzeitfehler erweiterbar ist.

In Tafel 1 ist als Zusammenfassung der Anforderungen und Möglichkeiten einer nutzereigenen Laufzeitfehlerbehandlung die An-

wenderdokumentation für den Bibliotheksmodul ERROR.BIB angegeben.
Die Realisierung dieses Moduls für Turbo-Pascal 3.00 unter CP/M (Z 80-CPU) ist in Bild 1 gezeigt. Dieser Modul wurde (der Kürze und Schnelligkeit wegen) zum Teil in INLINE-Code geschrieben. Hierbei gibt es gegenüber der obigen Dokumentation die folgende (unwesentliche) Einschränkung:
Bei mehrfach ineinandergeschachtelten Funktionen (function) ist (wegen ihrer linearen Anordnung im Speicher) nur eine Eindringtiefe depth=1 möglich.

Beispiel

Zur Veranschaulichung der Anwendung von ERROR.BIB soll in Bild 2 die Implementierung zweier Hyperbelfunktionen für eine häufig benötigte Bibliothek von Elementarfunktionen angegeben werden.

Schlußbemerkungen

Das vorgestellte Konzept zur Implementation und Anwendung einer nutzereigenen Laufzeitfehlerbehandlung bei Arithmetikfehlern ist nicht nur auf andere Compiler übertragbar, sondern auch auf andere Laufzeitfehlergruppen erweiterbar. Insbesondere gilt für wissenschaftlich-technische Berechnungen:
Bei Erweiterung der Fehlerbehandlung in einem Sprach-Compiler (für wissenschaftlich-technische Berechnungen) auf das vorgeschlagene nutzereigene Fehlerbehandlungskonzept bei Arithmetikfehlern wird die Implementation von mathematischer Software auf komfortable Weise ermöglicht. Die Software

- kommt ohne A-priori-Berücksichtigung von Rechnerinterna aus (kleinste und größte im Rechner darstellbare Zahl; Rechnergenauigkeit)
- ermöglicht Fehlermeldungen und -lokalisation beim primären Fehlerverursacher und damit an der Nutzeroberfläche.

Literatur
/1/ Schmidt, H.-J.: TURBO-PASCAL ohne Absturz. mc (1986) 6, S. 89
/2/ Gieselmann, K.: Eigene Fehler-Behandlung in TURBO-PASCAL-Programmen. Pascal (1987) 9, S. 94

✉ KONTAKT ☏
Technische Hochschule Ilmenau, Sektion MARÖK, Am Ehrenberg, Ilmenau, 6300; Tel. 7 42 42

Basic-Dateien unter Turbo-Pascal

Dr. Rainer Gebhardt
Forschungsinstitut für Textiltechnologie Karl-Marx-Stadt
Dr. Karl-Heinz Eger
Technische Universität Karl-Marx-Stadt, Sektion Mathematik

Die Möglichkeiten des Austausches von Informationen in Form von Dateien im Betriebssystem SCP sind sehr vielfältig. Die einfachste und wohl mit den geringsten Problemen behaftete Art des Austausches ist eine Übergabe im ASCII-Format. Für verschiedene Basissoftware werden dazu grundlegende Probleme in /1/ behandelt.
Berücksichtigt man, daß vor der Bereitstellung von Turbo-Pascal auf Personalcomputern Basic-80 eine der vorrangig benutzten Programmiersprachen war, so ist es verständlich, daß für viele Nutzer der Wunsch nach Weiterverwendung von Programmen und Daten in Basic auch unter Turbo-Pascal besteht.
Eine Änderung in fertigen Programmpaketen kann oftmals sehr problematisch sein und zu Fehlern führen. Da die in Basic verwendeten Dateien in der Regel Direktzugriffsdateien von Real-Zahlen sind, ergibt sich die Forderung, auch unter Turbo-Pascal diesen Dateityp zu lesen und zu schreiben. Die Möglichkeiten des Lesens und Schreibens von Direktzugriffsdateien in Basic werden in Bild 1 und Bild 2 angegeben. Konkret werden dabei die Zahlen von 1 bis zu einer angebbaren Zahl *n* geschrieben.
Ein Problem bei Verwendung dieser Basic-Dateien unter Turbo-Pascal besteht darin, daß im Normalfall eine Turbo-Pascal-Datei mit 4 Vorbyte beginnt, in denen die Anzahl und Länge der Datensätze binär verschlüsselt sind. Basic-Dateien besitzen diese Vorbytes nicht. Ein weiteres Problem besteht in der internen Darstellung von Real-Werten, deren Länge in Basic 4 Byte für einfache und 8 Byte für doppelte Genauigkeit beträgt, in Turbo-Pascal jedoch 6 Byte. Hinzu kommt noch die unterschiedliche Lage des Exponenten. Während in Basic der Exponent der Mantisse folgt, beginnt in Turbo-Pascal eine Real-Zahl mit dem Exponenten. Eine Gegenüberstellung der internen Darstellung der Real-Zahlen 1 bis 10 wird in Bild 3 gezeigt.
Vereinbart man in Turbo-Pascal die zu lesende Datei als untypisiertes File, so können mit den Prozeduren BlockRead und BlockWrite entsprechende Schreib-/Lese-Zugriffe ausgeführt werden (siehe /2/, S. 459, und /3/, S. 319). Zu beachten ist dabei, daß immer ein Block von 128 Byte gelesen bzw. geschrieben wird. Es werden also bei der Verwendung von Real-Zahlen einfacher Genauigkeit jeweils 32 Zahlen gelesen oder geschrieben.
Bild 4 zeigt ein Programm in Turbo-Pascal zum Lesen einer Datei von Real-Zahlen (4 Byte) im Basic-Format.
Zu bemerken ist dabei, daß die Anzahl von gelesenen Daten ein ganzzahliges Vielfaches von 32 beträgt, da unter SCP immer ein Block von 128 Byte geschrieben wird. Die Anzahl der tatsächlich zu lesenden Daten sollte daher bekannt sein oder beispielsweise als erste Zahl mit der Datei übergeben werden. Mit dem angegebenen Programm können so mittels Basic-Programmen erzeugte Dateien direkt unter Turbo-Pascal eingelesen und weiterverarbeitet werden.
Existieren für verschiedene Ausgabegeräte wie Drucker, Plotter und anderes bereits Aufbereitungsprogramme für Daten in Basic, so kann es wünschenswert sein, diese Programme unverändert zu nutzen. In diesem Fall muß eine Übergabe der Daten im internen Basic-Format erfolgen. Ein Programm unter Turbo-Pascal zur Ausgabe von Real-Zahlen im Basic-Format mit einfacher Genauigkeit wird in Bild 5 angegeben. Dabei wird der letzte zu schreibende Block gegebenenfalls mit Nullen aufgefüllt.
Es ist anzumerken, daß es beim Schreiben von Basic-Real-Zahlen unter Turbo-Pascal zu Rundungsfehlern kommt, da eine Beschneidung der Mantisse von 5 Byte auf 3 Byte erfolgt. Es sind demzufolge nur die ersten 6 Ziffern der Zahlen identisch. Will man

```
10 'Programm: Eingabe einer Direktzugriffsdatei
20 '          Real-Zahlen, einfache Genauigkeit
30 '
40 PRINT:LINE INPUT"Dateiname:";DATEI$
50 '
60 OPEN "R",#1,DATEI$,4    'Eroeffnen Datei
70 FIELD #1,4 AS A$        'Pufferzuweisung
80 '
90 SZE%=0                  'Satzzaehler
100 'Einlesen der Zahlen
110 GET #1
120 IF EOF(1) THEN 170
130 SZE%=SZE%+1            'Zaehler erhoehen
140 B=CVS(A$)              'Konvertieren Zeichenkette
150 PRINT B                'Ausgabe an den Bildschirm
160 GOTO 110
170 PRINT"Anzahl der Saetze =";SZE%
180 CLOSE #1
190 END
```

***Bild 1** Lesen einer Direktzugriffsdatei in Basic*

```
10 'Programm: Ausgabe einer Direktzugriffsdatei
20 '          Real-Zahlen, einfache Genauigkeit
30 '
40 PRINT:LINE INPUT"Dateiname:";DATEI$
50 INPUT"Anzahl Daten:";N%
60 '
70 OPEN "R",#1,DATEI$,4    'Eroeffnen Datei
80 FIELD #1,4 AS A$        'Pufferzuweisung
90 '
100 FOR I%=1 TO N%
110 LSET A$=MKS$(I%)       'Pufferspeicher fuellen
120 PRINT I%
130 PUT #1                 'Ausgabe
140 NEXT I%
150 CLOSE #1               'Schlieszen der Datei
160 END
```

***Bild 2** Schreiben einer Direktzugriffsdatei in Basic*

ZAHL	BASIC	TURBO-PASCAL
1	00 00 00 81	81 00 00 00 00 00
2	00 00 00 82	82 00 00 00 00 00
3	00 00 40 82	82 00 00 00 00 40
4	00 00 00 83	83 00 00 00 00 00
5	00 00 20 83	83 00 00 00 00 20
6	00 00 40 83	83 00 00 00 00 40
7	00 00 60 83	83 00 00 00 00 60
8	00 00 00 84	84 00 00 00 00 00
9	00 00 10 84	84 00 00 00 00 10
10	00 00 20 84	84 00 00 00 00 20

***Bild 3** Gegenüberstellung der internen Real-Zahlen-Darstellung in Basic und Turbo-Pascal*

```
program WRITEBAS;
        {Schreiben einer unter BASIC lesbaren Datei}
        {Real-Zahlen = 4 Byte }

var  f       : File;
     i,j,n,k: Integer;
     y       : Real;
     x       : Array[1..128] of Byte;
     datei   : String[14];

begin
    write('Dateiname :');readln(datei);
    assign(f,datei);
    rewrite(f);
    write('Anzahl der Daten :');readln(n);
    for i:=1 to n do
     begin
       y:=i;k:=(i-1) mod 32;
       x[k*4+4]:=mem[addr(y)];
       for j:=1 to 3 do
       x[k*4+j]:=mem[addr(y)+j+2];
       if (k=31) or (i=n) then     {Voller Block Schreiben}
           begin
             if i=n then           {Letzten Block mit 0 fuellen}
               for j:=4*(k+1)+1 to 128 do x[j]:=0;
             blockwrite(f,x,1);
           end;
     end;
    close(f);
 end.
```

***Bild 4** Lesen einer Basic-Datei unter Turbo-Pascal*

```
 program READBAS;
        {Lesen einer unter BASIC abgespeicherten Datei}
        {Real-Zahlen = 4 Byte }

var  f     : File;
     i,j   : Integer;
     y     : Real;
     x     : Array[1..128] of Byte;
     datei: String[14];

begin
    write('Dateiname :');readln(datei);
    assign(f,datei);
    reset(f);
    while not eof(f) do
     begin
      Blockread(f,x,1);
       for i:=1 to 32 do
        begin
         mem[addr(y)]:=x[(i-1)*4+4];
         mem[addr(y)+1]:=0;
         mem[addr(y)+2]:=0;
         for j:=0 to 2 do
         mem[addr(y)+j+3]:=x[(i-1)*4+j+1];
         writeln(y);
        end;
      end;
     close(f);
 end.
```

***Bild 5** Schreiben einer Basic-Datei unter Turbo-Pascal*

```
{BASIC-Real4-Zahl-Write-Read-Proceduren(BAS-W-R.INC)}
{** Dr. K.-H. EGER/Dr. R. GEBHARDT/ 16.06.1988 **}

Type UntypFile = File;

Procedure B_Write(Var f:UntypFile;i:Integer;x:Real);
     {Konvertieren einer 6-Byte-TURBO-Real-Zahl und}
     {Schreiben als i-te 4-Byte-BASIC-Real-Zahl in }
     {eine BASIC-Datenfile                          }
     Var b : Array[0..127] Of Byte;
         y : Array[0..3] Of Byte;
         z : Array[0..5] Of Byte Absolute x;
         j,k,fs : Integer;
     Begin
         fs:=FileSize(f);k:=i DIV 32;
         If k<fs Then Begin Seek(f,k);
                            BlockRead(f,b,1);
                      End
                 Else Begin FillChar(b,128,0);
                            Seek(f,fs);
                            For j:=fs To k Do
                            BlockWrite(f,b,1);
                      End;
         y[3]:=z[0];
         Move(z[3],y[0],3);
         Move(y,b[(i MOD 32)*4],4);
         Seek(f,k);
         BlockWrite(f,b,1);
     End;(*B_Write*)


Procedure B_Read(Var f:UntypFile;i:Integer;Var x:Real);
     {Lesen der i-ten 4-Byte-BASIC-Real-Zahl eines}
     {BASIC-Datenfiles und Konvertierung in eine  }
     {6-Byte-TURBO-Real-Zahl                      }
     Var b : Array[0..127] Of Byte;
         y : Array[0..3] Of Byte;
         z : Array[0..5] Of Byte Absolute x;
     Begin
         Seek(f,i DIV 32);
         BlockRead(f,b,1);
         Move(b[i*4 MOD 128],y,4);
         z[0]:=y[3];
         z[1]:=0;z[2]:=0;
         Move(y[0],z[3],3);
     End;{BlockRead}
```

***Bild 6** Quelltext des Include-Files BAS-W-R.INC*

```
{Programm zur Anwendung von BAS-W-R.INC}
{$I bas-w-r.inc}

var  f     :untypfile;
     datei:string[14];
     n,k,i:integer;
     jo    :char;
     x     :real;
label m1;
begin
     write('Dateiname :');readln(datei);
     assign(f,datei);{$I-}reset(f);{$I+}
     if ioresult>0 then rewrite(f);
m1:  write('Lesen/Schreiben/Ende: ');
     read(kbd,jo);writeln;
     if (jo='L') or (jo='l') then
        begin
          write('LESEN>Satz: ');read(n);
          b_read(f,n,x);
          writeln(' ----->',x);
          goto m1;
        end;
     if (jo='S') or (jo='s') then
        begin
          write('SCHREIBEN>Satz: ');read(n);
          write('    Wert: ');Readln(x);
          b_write(f,n,x);
          goto m1;
        end;
    close(f);
    writeln('ENDE');
 end.
```

***Bild 7** Anwendungsbeispiel für das Include-File BAS-W-R.INC* ▸

aus einem Turbo-Pascal-Programm heraus zu jeder beliebigen Zeit auf eine Basic-Datei zugreifen, so ist das mit einem gewissen Organisationsaufwand verbunden. Aus diesem Grund erweist es sich als sinnvoll, zwei Prozeduren **B_WRITE** und **B_READ** zu definieren, die diese Organisation übernehmen.

In der Prozedur **B_WRITE** werden 3 Parameter übergeben:

- f : Datei, auf die geschrieben werden soll
- i : Satznummer, unter der die Zahl abgelegt werden soll (1. Satz = 0) und
- x : Real-Zahl, die geschrieben werden soll.

Ebenso sind bei der Prozedur **B_READ** 3 Parameter anzugeben:

- f : Datei, von der gelesen werden soll,
- i : Satznummer, von der die Zahl gelesen werden soll (1. Satz = 0) und
- x : gibt die gelesene Real-Zahl zurück.

Beide Prozeduren sind aus dem im Bild 6 dargestellten Include-File ersichtlich und können in jedes beliebige Turbo-Pascal-Programm eingefügt werden. Dabei muß natürlich vorausgesetzt werden, daß eine entsprechende Datei mit **Assign** und **Reset** bzw. **Rewrite** vorbereitet wurde. Bei dem Versuch, einen Satz nach dem Ende des Files zu lesen, wird ein Laufzeitfehler erzeugt. Die beschriebenen Prozeduren sind auch auf dem A 7100 lauffähig, da gegenüber den Programmen in Bild 4 und Bild 5 auf **MEM[ADDR()]** verzichtet wurde.

Eine einfache Möglichkeit der Einbindung des Include-Files BAS_W_R.INC in ein Turbo-Pascal-Programm wird in Bild 7 gezeigt. Abschließend sei noch vermerkt, daß eine Übertragung der durchgeführten Überlegungen auf Real-Zahlen mit doppelter Genauigkeit (8 Byte) und Integer-Zahlen (2 Byte) einfach möglich ist.

Literatur

/1/ Müller, K.; Lennartz, M.: Arbeit mit ASCII-Dateien im Betriebssystem SCP. Mikroprozessortechnik 2 (1988) 5, S. 133

/2/ Ciric, Thies: Turbo Pascal systematisch. Band 1. München: te-wi Verlag 1987

/3/ Joepgen, H.-G.: Turbo-Pascal. München: Carl Hanser Verlag 1985

✉ KONTAKT ☎

Forschungsinstitut für Textiltechnologie, Abt. PM, PF 243, Karl-Marx-Stadt, 9010

Mikroprozessorsystem K 1810 WM 86

Hardware · Software · Applikation (Teil 7)

Prof. Dr. Bernd-Georg Münzer
(wissenschaftliche Leitung),
Dr. Günter Jorke, Eckhard Engemann,
Wolfgang Kabatzke, Frank Kamrad,
Hellfried Schumacher, Tomasz Stachowiak
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik,
Wissenschaftsbereich Mikrorechentechnik/
Schaltungstechnik

8.6. Systemschaltkreise

Die CPU 80286 ist ein VLSI-Schaltkreis mit etwa 130000 integrierten Transistoren im 68poligen Gehäuse.
Adreß- und Datenbus des Prozessors sind getrennt herausgeführt, belegen also 24 bzw. 16 Anschlüsse. Weitere 17 Pins führen Steuer- und Statussignale, ähnlich denen des 8086. Die restlichen Anschlüsse sind entweder nicht beschaltet oder dienen zur Spannungsversorgung. Die 80286-CPU benötigt eine Betriebsspannung von 5 V. Alle Ein- und Ausgänge sind TTL-kompatibel.
Zum System 80286 gehören außerdem der Taktgenerator 82284 und der Buscontroller 82288, deren Funktionen denen der entsprechenden 8086-Systemschaltkreise ähneln.
Der dazugehörige Arithmetik-Koprozessor 80287 ist code-kompatibel zum 8087. Ein Unterschied besteht jedoch in der Einbindung des Arithmetikprozessors in das System 80286. Der 80287 ist eigentlich ein Peripherieschaltkreis, das heißt, der Datentransfer mit der CPU wird über Ein- und Ausgabezyklen realisiert, wofür der E/A-Adreßbereich von 00F8H bis 00FFH reserviert ist. Für den Datentransfer werden die E/A-Buszyklen in der CPU automatisch ausgelöst.
Als Interface-Schaltkreise können im System 80286 die gleichen Typen wie im System 8086 verwendet werden, wobei aber auf die Einhaltung der Zeitbedingungen zu achten ist.
Weiterhin gibt es einige Koprozessor- und Controller-Schaltkreise, die über eine spezielle 80286-Betriebsart verfügen und so als 80286-Systemschaltkreise verwendet werden können. Dazu gehören die dRAM-Controller 8207 und 8208, der DMA-Koprozessor 82258 und der Grafik-Koprozessor 82786.

9. Multitaskverarbeitung

9.1. Grundprinzipien der Echtzeitverarbeitung

Für 8086-Rechner existieren neben Einzelnutzerbetriebssystemen (z.B. SCP 1700, CP/M 86, DCP, MS-DOS) auch Echtzeitbetriebssysteme (z.B. BOS 1810, RMX86, EMOS, RMOS2).
Echtzeitbetriebssysteme verfügen im allgemeinen über eigene Bestandteile für die Entwicklung von Programmen auf dem Assemblerniveau oder mit Hochsprachen (Editier-, Übersetzer-, Link- und Debuggprogramme). Diese benutzen eine eigene Dateiorganisation, die durch eine Vielzahl von Dienstprogrammen unterstützt wird.
Bei der Programmentwicklung mit einem Echtzeitbetriebssystem werden Subsysteme des Betriebssystems in das Anwenderprogramm übernommen, wodurch sich neue Formen der Programmorganisation ergeben. Im folgenden sollen deshalb nicht die Hilfsmittel der Programmentwicklung mit einem Echtzeitbetriebssystem, sondern die Eigenschaften des echtzeitfähigen Anwenderprogramms im Mittelpunkt der Darlegungen stehen.
Dafür werden Anwenderprogramme betrachtet, die das Subsystem **NUCLEUS** (deutsch: Kern) des Echtzeitbetriebssystems BOS 1810/1/ einschließen.
Verschiedene Möglichkeiten für die Bildung einer solchen Programmkonfiguration werden am Ende dieses Abschnitts aufgezeigt.
Die charakteristischen Eigenschaften echtzeitfähiger Programme liegen in der quasiparallelen Abarbeitung einzelner Programmteile, der **Tasks**, und der Unterstützung der **Interruptverarbeitung.**
Die Formulierung eines Anwenderprogramms aus verschiedenen Tasks für die Behandlung unterschiedlicher Ereignisse in einem Prozeß führt zu einer weitgehenden Entflechtung von Teilprogrammen und zu einer hohen Flexibilität bei der Programmentwicklung und bei Änderungen.
In einem Multi-Task-Programm wird die Abarbeitung der Tasks von einem Task-Scheduler (deutsch: Ablaufauflister) innerhalb des NUCLEUS nach dem Prioritätsprinzip gesteuert.
Tasks werden vom NUCLEUS als **Objekte** verwaltet. Weitere Objekttypen dienen der Ressourcenverwaltung des Rechners (Job), der dynamischen Speicherzuweisung (Segment), dem Informationsaustausch zwischen den Tasks (Mailbox), der Synchronisation der Taskabarbeitung (Semaphore und Region).
Jede Task erhält einen Prioritätswert, einen der Ausführungszustände: *rechenbereit* (ready), *schlafend* (asleep) oder *suspendiert* (suspended), einen eigenen Stackbereich und meist einen eigenen Datenbereich.
Die jeweils höchstpriorisierte, rechenbereite Task wird abgearbeitet. In den Wartezeiten der höchstpriorisierten Task übernimmt die nächst höher priorisierte, rechenbereite Task den Prozessor.
Wartezeiten können prozeßbedingt (z.B. bei E/A-Operationen) auftreten oder durch eine Vielzahl von Systemrufen ausgelöst werden, so daß auch niedrig priorisierte Tasks in die Abarbeitung kommen.
Der Objekttyp **Job** umfaßt ein Kontingent an Rechnerressourcen und Objekten. Unter den Tasks eines Jobs befindet sich mindestens eine Jobinitialtask. Durch die Bildung verschiedener Jobs kann der Arbeitsbereich des Rechners in voneinander abgegrenzte Bereiche aufgeteilt werden. Eine Task kann innerhalb eines Jobs dessen Ressourcen auf weitere Jobs („Kinderjobs") aufteilen.
Die Gesamtressourcen des Rechners werden von einem Wurzeljob (rootjob) verwaltet, der vom NUCLEUS erzeugt wird und dessen Initialtask (Wurzeltask) nach dem Anlauf des Systems gestartet wird.
Innerhalb der Wurzeltask werden ein oder mehrere Anwenderjobs gebildet, die die Anwendertasks enthalten.
Bild 9.1 zeigt die Struktur eines Multi-Task-Programms. Am Anfang werden in einem Initialisierungsteil die Hardwareeinstellungen (Programmierung Timer und Interruptcontroller) vorgenommen. Im NUCLEUS werden die internen Systemdaten für die Verwaltung aller Objekttypen und der Wurzeljob erzeugt.
Innerhalb eines Anwenderjobs werden Tasks kreiert, die auf die Ressourcen des Jobs zugreifen können.
Die Reaktionsfähigkeit eines Echtzeitprogramms auf externe Ereignisse wird durch die Bildung von **Interrupttasks** gewährleistet. Eine Interrupttask wird durch eine Interruptanforderung aus der Hardware in den rechenbereiten Ausführungszustand überführt.

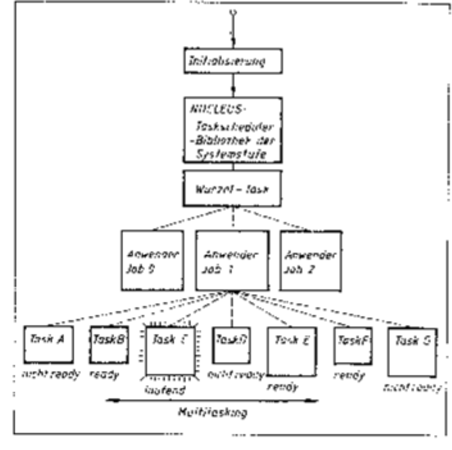


Bild 9.1 Struktur eines Multitaskprogramms

9.2. Task-, Job- und Segment-Objekte

Der NUCLEUS enthält eine Bibliothek von Systemrufen, die in Anwendertasks aufgerufen werden können. Die Maximalversion des NUCLEUS von 24 KByte verfügt über 56 Systemrufe. Kleinere NUCLEUS-Versionen mit einem Subset von Systemrufen lassen sich im Konfigurationsprozeß festlegen. Alle Objekte werden durch Systemrufe erzeugt und gelöscht.

9.2.1. Erzeugen von Tasks

Nach der Vorgabe einer Priorität (Wert 0 - 255; 0: höchste Priorität), der Startadresse des Task-Programms, eines Wertes für das

Datensegment, des Taskstackpointers, der Stackgröße, des Taskflags für die Angabe der 8087-Koprozessornutzung und eines Pointers für die Ablage des Fehlercodes im Stack der aufrufenden Task, erzeugt der Systemruf **CREATE_TASK** eine Task für das angegebene Programm. Dabei entstehen die Task-Systemdaten, das heißt ein 80 Byte großer Steuerblock mit den Task-Informationen.
Wie jedes Objekt wird eine Task durch einen **Token,** einen 16-Bit-Idendifikator, gekennzeichnet. Der Token enthält zugleich die Adreßinformation der Task-Systemdaten.
Eine neu erzeugte Task besitzt den Ausführungszustand *rechenbereit.*
Der Task-Scheduler erzeugt eine Verkettung aller rechenbereiten Tasks durch die Aufnahme der Token der nächst höher und nächst niedriger priorisierten Tasks in die Systemdaten einer jeden Task.
Eine zweite gelinkte Liste von Task-Systemdaten existiert für die nicht rechenbereiten Tasks. Beim Bilden und Löschen von Tasks oder bei der Änderung des Ausführungszustandes von Tasks werden die Listen neu geordnet.
Der Systemruf **DELETE_TASK** mit der Vorgabe des Task-Tokens im Stack löscht die Task.

9.2.2. Erzeugen von Jobs

Der Objekttyp *Job* verwaltet einen Speicherbereich (memory pool) und eine Menge von Objekten.
Ein Job enthält einen Katalog (Jobverzeichnis, job directory), in dem für Objekte die Zuordnung von Namen zu den Objekttoken eingetragen werden kann. Da für den Zugriff auf jedes Objekt der Token benötigt wird, bietet der Katalog die Möglichkeit, die Token von in anderen Tasks erzeugten Objekten zu bestimmen. Bei der Bildung eines Jobs mit dem Systemruf **CREATE_JOB** werden u.a. die Größe des Jobverzeichnisses, der Speicherbedarf, die Maximalzahl der Objekte, die Maximalzahl der Tasks und alle Parameter für die Initialtask vorgegeben. Ein erzeugter Job wird durch einen Jobtoken gekennzeichnet.
Der NUCLEUS vergibt den von den Objekten beanspruchten Speicher mit fallender Adresse.
Nach Bild 9.2 wird bei der Erzeugung eines Jobs zuerst das Jobobjektverzeichnis an der Obergrenze des noch verfügbaren Arbeitsspeichers angelegt. Daran schließen sich die Jobsystemdaten an.
Für die mit dem Job erzeugte Initialtask wird zunächst der zugeordnete Stackbereich reserviert. Darunter liegen die Task-Systemdaten.
Der Systemruf **DELETE_JOB** löscht einen Job.

9.2.3. Erzeugen von Segmenten

Der innerhalb des umgebenden Jobs verfügbare Arbeitsspeicher kann von einer Task portionsweise angefordert werden. Der Systemruf **CREATE_SEGMENT** mit der Vorgabe der Speichergröße erzeugt ein Segmentobjekt. Segmente enthalten z.B. den Programmcode für die Tasks (Bild 9.2.). Der zugewiesene Speicherbereich wird mit dem Systemruf **DELETE_SEGMENT** wieder freigegeben.

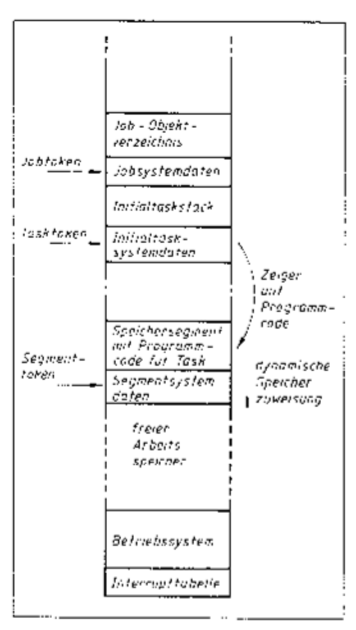


Bild 9.2 dynamische Speicherzuweisung bei der Bildung von Objekten

9.2.4. Taskumschaltung

Das dem BOS 1810-NUCLEUS zugrunde liegende Prioritätsprinzip setzt voraus, daß eine in der Abarbeitung befindliche Task selbständig den Prozessor, zumindest zeitweise, freigibt.
Zu den einfachsten Möglichkeiten zählen die Systemrufe für die zeitweilige Prozessorfreigabe durch die aktive Task.
Der Systemruf **SLEEP** versetzt die angegebene Task (meist die aktive) für eine vorgegebene Zeit in den Abarbeitungszustand *schlafend.* Nach dem Ablauf dieser Zeit erhält die Task automatisch wieder den Abarbeitungszustand *rechenbereit.* Der Systemruf **SUSPEND_TASK** suspendiert die aktive oder eine andere Task von der Abarbeitung. Durch mehrfaches Suspendieren wird eine entsprechende Suspendierungstiefe eingestellt.
Eine suspendierte Task kann von der aktiven Task mit dem **RESUME_TASK**-Ruf wieder in einen *rechenbereiten* Abarbeitungszustand gebracht werden. Für eine mehrfach suspendierte Task sind mehrere RESUME_TASK-Rufe notwendig.

9.2.5. Systemrufe für den Objektzugriff

Für Informationen über die Objekte steht eine Anzahl von Systemrufen zur Verfügung.
Über den Systemruf **GET_TASK_TOKENS** können die Token der laufenden Task, des sie enthaltenden Jobs oder des Wurzeljobs bestimmt werden.
Die Priorität einer Task wird mit dem Systemruf **GET_PRIORITY** ermittelt.
Der Token eines beliebigen Objektes kann unter einem bis zu 12stelligen Namen in das Verzeichnis eines Jobs mit dem Ruf **CATALOG_OBJECT** ein- und mit dem Ruf **UNCATALOG_OBJECT** ausgetragen werden.
Für katalogisierte Objekte ergibt der Ruf **LOOKUP_OBJECT** bei Vorgabe des Namens den Token.

9.3. Task-Kommunikation und -Synchronisation

9.3.1. Informationsaustausch zwischen den Tasks

Die weitgehende Unabhängigkeit in der quasiparallelen Abarbeitung der einzelnen Tasks setzt spezielle Techniken für die Informationsübermittlung zwischen den Tasks voraus.
Diesem Zweck dienen Mailboxobjekte. Eine Mailbox besteht nach Bild 9.3 aus zwei Warteschlangen. In der Objektwarteschlange können Nachrichten abgespeichert werden. Für eine Nachricht steht in der Objektwarteschlange der Token eines beliebig großen Segmentes mit den zu übertragenden Informationen. Die Segmenttoken können von allen Tasks, die eine Zugriffsmöglichkeit auf die Mailbox besitzen, eingeschrieben werden. Für den Zugriff auf eine Mailbox benötigt eine Task lediglich deren Token.

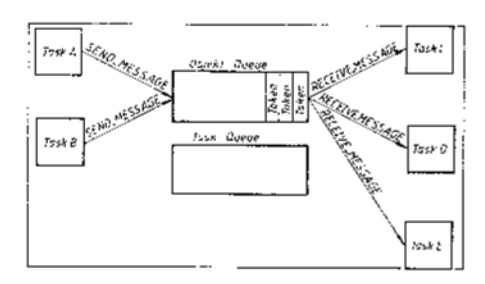


Bild 9.3 Objekttyp Mailbox

Mit der Kenntnis des Mailboxtokens kann eine Task von einer Mailbox auch Nachrichten anfordern. Falls mehr Nachrichtenanforderungen als Segmenttoken für eine Mailbox vorliegen, entsteht eine Warteschlange von Tasks.
Eine Mailbox wird von einer Task mit dem **CREATE_MAILBOX**-Ruf erzeugt. Dabei kann die Organisationsform und Größe der Warteschlangen angegeben werden. Die Bekanntgabe des Tokens für andere Tasks kann z.B. durch Katalogisieren im Objektverzeichnis eines übergeordneten Jobs erfolgen.
Der Token des Segmentes, das die Information enthält, kann mit dem **SEND MESSAGE**-Systemruf in eine Mailbox geschrieben werden.
Mit dem Ruf **RECEIVE_MESSAGE** wird eine Nachricht von der Mailbox gelesen. Eine Leseanforderung an eine Mailbox, die keine Nachricht enthält, versetzt die anfordernde Task in den Abarbeitungszustand *schlafend.* In diesem Fall gibt die anfordernde Task den Prozessor frei, bis in der Mailbox eine Nachricht eingegangen ist. Dadurch kommt die anfordernde Task wieder in den *rechenbereiten* Abarbeitungszustand.
Der Systemruf **DELETE_MAILBOX** löscht eine Mailbox.

9.3.2. Taskumschaltung über Semaphore

Der Objekttyp **Semaphore** (Zeichenträger) dient der Synchronisation der Taskabarbeitung. Durch die Anforderung von *Einheiten* (units) des Kontingentes eines Semaphores mit dem Systemruf **RECEIVE_UNITS** kann die

Abarbeitung einer Task bedingt unterbrochen werden. Falls die Anzahl der angeforderten Einheiten nicht in dem Semaphore enthalten ist, geht die anfordernde Task in den Zustand *schlafend*, und der Token der anfordernden Task wird in die Taskqueue eingetragen. Nach der Zuführung von Einheiten an das Semaphore mit dem Systemruf **SEND_UNITS** durch eine aktive Task wird die an der Mailbox wartende Task wieder *rechenbereit*.
Nach Bild 9.4 enthält ein Semaphore eine Taskqueue, in der alle anfordernden Tasks nach einem einstellbaren Organisationsprinzip (FIFO oder Prioritätsordnung) abgespeichert werden, falls das Kontingent an Einheiten für die erste anfordernde Task nicht ausreicht.
Semaphore können mit den Systemrufen **CREATE_SEMAPHORE** und **DELETE_SEMAPHORE** erzeugt und gelöscht werden.

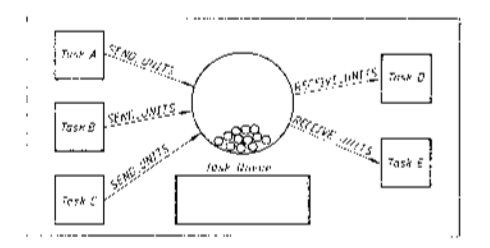


Bild 9.4 Objekttyp Semaphore

9.3.3. Datenzugriffssteuerung mit Regionen
Der ungestörte Zugriff auf Daten (oder die Abarbeitung von Teilprogrammen) kann durch Objekte vom Typ **Region** geschützt werden.
Der Zugriffsschutz wird durch eine Task mit dem Systemruf **RECEIVE_CONTROL** an die Region angefordert. Eine höherpriorisierte Task, die ebenfalls den **RECEIVE_CONTROL**-Systemruf an die Region enthält, kann auf die gemeinsamen Daten erst zugreifen, wenn diese von der vorherigen Task vervollständigt worden sind.
Da bereits die Anforderung der vorherigen Task in der Taskqueue der Region eingetragen ist, geht die höherpriorisierte Task in den Zustand *schlafend*, und die vorherige Task setzt die Abarbeitung fort. Die nun wartende Task wird in die Taskqueue der Region eingetragen.
Der Systemruf **SEND_CONTROL** löscht die Zugriffsanforderung der aktiven Task in der Region.
Eine alternative Form der Zugriffsanforderung bietet ein weiterer Systemruf **ACCEPT_CONTROL**, die nur bedient wird, wenn keine früheren Anforderungen in der Taskqueue der Region vorliegen.
Regionobjekte werden mit den Systemrufen **CREATE_REGION** und **DELETE_REGION** erzeugt und gelöscht.

9.4. Echtzeitverarbeitung

Die schnelle Reaktion auf externe Ereignisse ist eine der Hauptaufgaben eines Echtzeitbetriebssystems. Dafür wird die im System 8086 enthaltene Interruptorganisation mit der Multitaskverarbeitung kombiniert.
Nach der im Abschnitt 4 beschriebenen Interruptorganisation muß die Bearbeitung für ein externes Ereignis durch ein Anforderungssignal an einem INTR-Eingang eines Slave- oder des Master-PIC (programmable interrupt controller) angefordert werden.
Die in der Hardware ausgewählte Anforderung mit der höchsten Interruptpriorität führt im Fall der Interruptfreigabe zur Unterbrechung jeder beliebig hoch priorisierten Task. Mit der eingeschobenen Interrupt-Service-Routine können die für das externe Ereignis benötigten Reaktionen sofort ausgelöst werden. Mit dieser Methode, die für die Bearbeitung dringender Fälle auch möglich ist, wird jedoch die Prioritätsorganisation der Multitaskverarbeitung übergangen.
Im allgemeinen Fall soll auch die Bearbeitung externer Ereignisse der Prioritätsentscheidung der Multitaskverarbeitung untergeordnet werden. Dafür werden **Interrupttasks** eingesetzt.
Eine Interrupttask wird wie jede Task mit einem CREATE_TASK-Systemruf erzeugt und durch ein zusätzliches Interruptprogramm, den Interrupthandler, in der Abarbeitung gesteuert. Durch den Systemruf **SET_INTERRUPT** innerhalb der Interrupttask wird diese einem INTR-Eingang eines Slave- oder des Master-PIC zugeordnet. Zugleich wird die Adresse des zugehörigen Interrupthandlers in die Interrupttabelle eingetragen. Der Systemruf bildet außerdem einen Puffer, in dem die Anforderungen für den entsprechenden INTR-Eingang abgespeichert werden.
Nach Bild 9.5 wird eine Interrupttask nach ihrer Bildung entsprechend der Taskpriorität gestartet. Nach der Interrupthandlerzuordnung versetzt der Systemruf **WAIT_INTERRUPT** die Interrupttask in den Zustand *schlafend*.
Eine durch ein externes Ereignis ausgelöste und an die CPU durchgeschaltete Interruptanforderung führt zur sofortigen Ausführung des zugehörigen Interrupthandlers. Diese enthält im wesentlichen nur den Systemruf **SIGNAL_INTERRUPT**, durch den die Interrupttask *rechenbereit* wird.

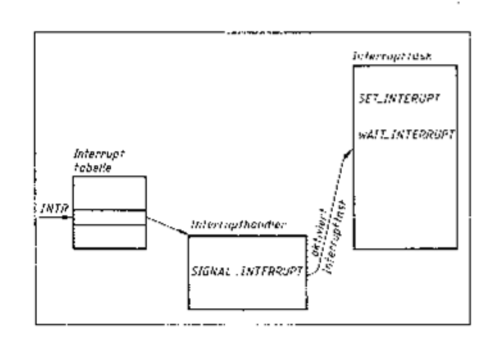


Bild 9.5 Interruptbehandlung mit Interrupttask

9.5. E/A-Operationen

In einem Echtzeitprogramm muß der Nutzung der bei E/A-Operationen auftretenden Wartezeiten besondere Aufmerksamkeit gewidmet werden.
Für E/A-Operationen existieren im Echtzeitbetriebssystem BOS 1810 die Subsysteme BIOS (basic I/O system) und EIOS (extended I/O system).
Für spezielle Zielrechnerkonfigurationen ist es oft günstiger, anwendungsspezifische E/A-Systeme in Anlehnung an die Organisationsprinzipien des BIOS zu entwickeln. Das Subsystem BIOS verwendet im Prinzip die im folgenden beschriebene Organisationsform.
Einheitlich für alle Geräte werden asynchrone Systemrufe für das Lesen (**READ**) und Schreiben (**WRITE**) von Daten benutzt. Der Aufruf einer E/A-Operation erzeugt einen E/A-Anforderungsblock (I/O request segment, IORS) mit allen E/A-Zugriffsinformationen. Die Anforderung wird in eine Warteschlange von Anforderungsblöcken (IORS queue) eingeordnet. Die Warteschlange entsteht nach Bild 9.6 durch vor- und rückwärts gerichtete Zeiger in jedem Block auf die benachbarten Blöcke. Nach dem Aufruf einer E/A-Operation kann die aufrufende Task die Abarbeitung sofort fortsetzen. Im allgemeinen Fall geht die aufrufende Task vor der Verwendung der angeforderten Daten mit einem RECEIVE_MESSAGE-Systemruf in den Zustand *schlafend* und gibt den Prozessor frei. Der Token der Mailbox, die die Verfügbarkeit der Daten nach der Datenübertragung meldet, wird beim E/A-Aufruf als Parameter angegeben.
Für jedes E/A-Gerät existiert je eine Warteschlange von E/A-Anforderungsblöcken.
Jede Warteschlange wird von einer hochpriorisierten Queuetask bedient. Die Queuetask übergibt bei Bereitschaft des E/A-Gerätes die Zugriffsparameter an den Gerätetreiber und gibt danach den Prozessor frei.
Das E/A-Gerät meldet die Übertragungsbereitschaft mit einem Interrupt. Dieser aktiviert eine Interrupttask, die die Datenübertragung zwischen dem Gerät und einem Pufferspeicher ausführt.
Die Interrupttask meldet den Abschluß der Datenübertragung an die Queuetask, die die Fertigmeldung an die Mailbox in der aufrufenden Task weitergibt und gegebenenfalls eine weitere E/A-Anforderung an den Gerätetreiber übergibt.
Die aufrufende Task wird durch die Fertigmeldung an die Mailbox wieder *rechenbereit*.
Nach diesem Prinzip können Anforderungen an alle E/A-Geräte asynchron von jeder Task und aus jedem Job gestellt werden. Die E/A-Subsysteme des BOS 1810 bieten zusätzlich Möglichkeiten der Vergabe von Zugriffsberechtigungen zu E/A-Geräten an die Jobs.

9.6. Echtzeitprogrammentwicklung

Die Entwicklung eines Echtzeitprogramms erfolgt im allgemeinen Fall mit der Entwicklungstechnik eines Echtzeitbetriebssystems. Im Echtzeitbetriebssystem BOS 1810 können die Systemrufe der Subsysteme in Assembler- und PL/M-86-Programmen benutzt werden. Mit Hilfe des Link-Programms wird eine Interfacebibliothek für den Anschluß der Systemrufe in das 8086-Maschinenprogramm eingebunden.
Echtzeitprogramme können jedoch auch auf anderen Betriebssystemen (SCP 1700, MS-DOS) auf dem Assemblerniveau oder in der Programmiersprache C entwickelt werden.

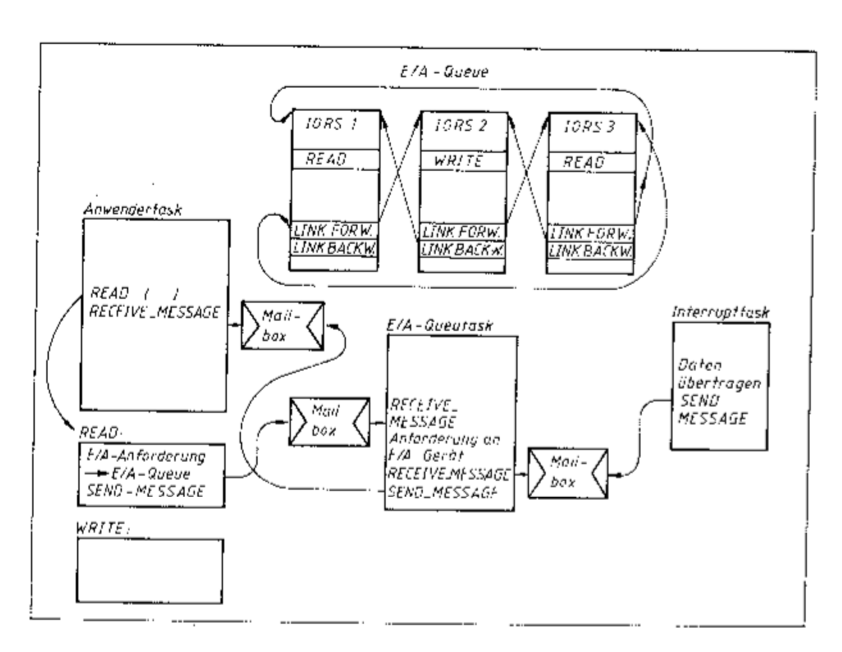


Bild 9.6 *Organisation der taskgesteuerten E/A-Operationen*

Eine Multitaskergänzung des Wirtsbetriebssystems steuert die Multitaskverarbeitung. Ein als Task laufendes spezielles Ladeprogramm (Applikationslader) für die Dateiformen der ausführbaren Programme (z. B. .CMD-Dateien) erzeugt die Tasks.
Dabei können die Anwenderprogramme auch die Systemrufe des Wirtsbetriebssystems nutzen.

9.7. Systemdebugger

Die Testung eines Echtzeitprogrammes erfordert spezielle Hilfsmittel für die Diagnose der Objekteigenschaften. Der Systemdebugger des Betriebssystems BOS 1810 enthält unter anderem die folgenden Kommandos für die Diagnose der im Echtzeitprogramm enthaltenen Objekte:

vj [<jobtoken>]
Darstellung der Jobhierarchie vom angegebenen Job an
vk
Angabe der Listen der *rechenbereiten* und *nicht rechenbereiten* Tasks
vo <jobtoken>
Angabe aller Objekte des Jobs
vd <jobtoken>
Angabe des Objektverzeichnisses des Jobs
vt
Angabe der Objektattribute

Auch ohne einen Systemdebugger können die Objektattribute aus der Analyse der Objekt-Systemdaten gewonnen werden.

9.8. Programmbeispiel

Bild 9.7 enthält einen Assemblerprogrammausschnitt eines Multitaskprogramms. Ein in der Assemblerprogrammliste enthaltenes Teilprogramm TSK1 wird mit dem Systemruf CREATE_TASK als Task kreiert. Der Anschluß des Systemrufes an das Subsystem NUCLEUS wird mit dem Unterprogramm NCRTSK vollzogen, das das zugehörige Interfacebibliotheksprogramm darstellen soll.
Wenn das laufende Programm den Prozessor freigibt, wird die erzeugte Task, falls keinen anderen höherpriorisierten Tasks *rechenbereit* sind, zur Ausführung gelangen, ohne daß ein CALL-Befehl ausgeführt wurde.

9.9. Objektorientierte Programmiertechniken

Die objektorientierte Programmierung findet in zunehmendem Maße in modernen höheren Programmiersprachen und für die Organisation der Parallelverarbeitung Anwendung. Die Erweiterungen beziehen sich vor allem auf die Definition einer Vielzahl anwenderspezifischer Objekttypen.
Bei der quasiparallelen oder echt parallelen Abarbeitung der Objekte (Architekturen mit mehreren Prozessoren) werden Programme und Daten von den übrigen Programmteilen vollständig isoliert. Informationen zwischen den Objekten werden nur über Nachrichten ausgetauscht.
Neben diesen Vorteilen kann die objektorientierte Programmierung in vielen Fällen für eine vereinfachte implizite Programmformulierung genutzt werden.
Im folgenden sollen diese Eigenschaften bei der Anwendung der NUCLEUS-Objekte demonstriert werden. Die dabei entstehenden schlechteren Abarbeitungseigenschaften

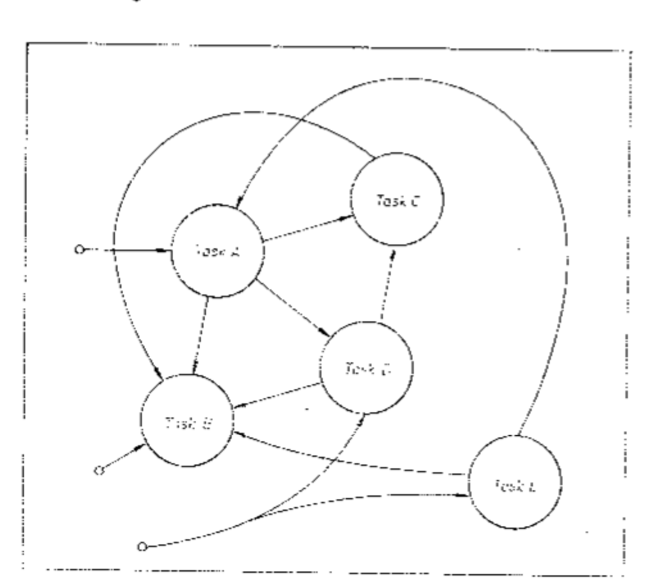
```
DATA    DSEG
TSKTOK  RW      1
PRTY    DW      200
STSIZE  DW      400H
ECODE   RW      1
        .
        .
CODE    CSEG
        .
        .                           ;Vorgabeparameter für Systemruf
        .                           ;CREATE_TASK in Stack schreiben:
        PUSH WORD PTR PRTY          ;Taskpriorität
        PUSH CS                     ;Programmadresse, Segmentanteil
        MOV  AX,OFFSET TSK1
        PUSH AX                     ;Programmadresse, Offsetanteil
        MOV  AX,0
        PUSH AX                     ;Wert für DS
        PUSH AX                     ;Stackpointerzuweisung durch
        PUSH AX                     ;NUCLEUS
        PUSH WORD PTR STSIZE        ;Stackgrösze
        PUSH AX                     ;Taskflag (keine 8087-Nutzung)
        PUSH DS                     ;Zeiger für Fehlercode
        MOV  AX,OFFSET ECODE
        PUSH AX
        CALL NCRTSK                 ;Interfacebibliotheksprogramm
        MOV  TSKTOK,AX              ;Tasktoken abspeichern
        .
        .                           ;Vorgabeparameter für Systemruf
                                    ;DELETE_TASK in Stack schreiben:
        MOV  AX,0
        PUSH AX                     ;laufende Task löschen
        PUSH DS                     ;Zeiger für Fehlercode
        MOV  AX,OFFSET ECODE
        PUSH AX
        CALL NDLTSK                 ;Interfacebibliotheksprogramm

TSK1:   .                           ;Assemblerprogramm für zu
        .                           ;erzeugende Task
        JMP  TSK1

                                    ;Eintrittspunkte in die Inter-
                                    ;facebibliothek für NUCLEUS-
                                    ;Systemcalls:
NCRTSK: .                           ;CREATE_TASK
        .
NDLTSK: .                           ;DELETE_TASK
        .
```

Bild 9.7 *Assemblerprogrammausschnitt mit der Bildung einer Task*

Bild 9.8 *Objektorientierte Programmiertechnik am Beispiel der Entflechtung eines Netzes*



(größere Verarbeitungszeit) im Vergleich zu konventionellen Lösungen sind auch für die objektorientierte Programmierung typisch.
Problemstellungen mit Signalflußcharakter nach Bild 9.8 können in die Bearbeitung von Teilproblemen zergliedert werden. Dabei ist es notwendig, daß für die Bearbeitung bestimmter Teilsysteme die Ergebniswerte anderer Teilsysteme vorliegen müssen. In konventionellen Programmlösungen muß deshalb zuerst die Berechnungsreihenfolge der Teilsysteme festgelegt werden.
In der objektorientierten Programmierung wird die Berechnung aller Teilsysteme quasiparallel mit je einer Task gestartet. Die Startreihenfolge ergibt sich aus den willkürlich zugeordneten Taskprioritäten. Alle für die Verkopplung der Teilsysteme benötigten Informationen werden mit NUCLEUS-Systemrufen über Mailboxen vermittelt.
Die Bearbeitung eines Teilproblems wird automatisch unterbrochen, bis dafür benötigte Ergebnisse von anderen Objekten verfügbar sind. Damit wird der Prozessor für die Bearbeitung anderer Teilsysteme freigegeben. Auf diese Weise stellt sich auf der Grundlage der Multitasksteuerung die notwendige Reihenfolge der Bearbeitung der Teilsysteme ein. Dieser Mechanismus kann auch für die Behandlung einer Dead-lock-Situation, in der mehrere Tasks gegenseitig auf Nachrichten warten, genutzt werden. In diesem Fall wird eine niedrig priorisierte Task aktiv, die diese Situation bedient.

10. Programmentwicklung in C

In diesem Kapitel wird der C-Compiler *DRC* vorgestellt. Ziel des Beitrages ist es, Hinweise für die Bedienung und Nutzung des C-Compilers DRC und die Implementation der Hochsprache **C** auf dem K 1810 WM86 (8086) unter dem Betriebssystem CP/M-86 zu geben. Dabei wird auf den bekannten Veröffentlichungen /1, 2, 3/ zur Anwendung der Programmiersprache C aufgebaut.

10.1. Systemprogramme für die Programmierung in C

Das Herangehen bei der Entwicklung von C-Programmen entspricht der für Assemblerprogrammierung in Abschnitt 6 beschriebenen Weise. Zunächst werden die Quellprogramme editiert und dann mit Hilfe des C-Compilers in Objektdateien *.obj übersetzt.
Nach dem Linken eines bzw. mehrerer Programme zu einem CMD-File wird das Programm durch Aufrufen des Namens gestartet.
Die Fehlersuche erfolgt günstig auf Maschinenniveau mit dem symbolischen Debugger *SID86* (unter Zuhilfenahme eines Reassemblerlistings) oder durch zweckmäßiges Einfügen von Ausgabeanweisungen in das Programm (z. B. Anzeige lokaler Variablen), die bei nachgewiesener Fehlerfreiheit des Programms wieder entfernt werden können.

10.1.1. Bestandteile und Arbeitsweise des C-Compilers DRC

Das gesamte C-Compilerprogrammpaket setzt sich aus den folgenden Teilprogrammen zusammen:

DRC.CMD	– Basismodul des C-Compilers
DRC860.CMD	– Präprozessor
DRC861.CMD	– Codegenerator
R.CMD	– Programm zum Nachladen des Codegenerators
DRC862.CMD	– Disassembler
DRC.ERR	– Fehlerbeschreibungsdatei
DRCRPP.CMD	– Rückübersetzer für Präprozessor

Zusätzlich sind die Laufzeitbibliotheken **CLEARS.L86** und **CLEARL.L86** sowie für bestimmte Anwendungen die Header-Dateien **STD.H**, **PORTAB.H**, **CTYPE.H** und **VAR.H** erforderlich.

Die Übersetzung eines C-Programms geht wie folgt vor sich: Zunächst wird das Quellprogramm vom Präprozessor vorübersetzt. Dabei werden die Präprozessoranweisungen wie Dateieinfügungen, Makrosubstitutionen und bedingte Generierungsanweisungen ausgeführt. Die entstehende Datei befindet sich auf dem aktuellen Laufwerk und hat den Namen CTEMP.TOK. Die für DRC860 möglichen Präprozessoranweisungen lauten:

#define
Makrodefinitionen mit und ohne Parameter. Eine Makrodefinition kann maximal 10 Parameter enthalten.
#undef
löscht eine Makrodefinition
#include<d:file>
Einfügen der Datei *file* vom angegebenen Laufwerk. Wurde kein Laufwerk angegeben, wird die Datei vom durch die –i-Option (siehe 8.1.3.) festgelegten Laufwerk geladen.
#include"d:file"
Einfügen der Datei *file* vom angegebenen Laufwerk. Wurde kein Laufwerk angegeben, wird das aktuelle Laufwerk benutzt.
#if, #ifdef, #ifndef, # endif, #else
bedingte Compilierung des folgenden Quelltextes
#nolist
Diese im C-Standard /1, 2/ nicht vorkommende Präprozessoranweisung bezieht sich auf die Generierung des Disassemblerlistings durch DRC862. Der nachfolgende Quelltext wird bei der Auflistung unterdrückt.
#list
Der nachfolgende Text wird in das Disassemblerlisting aufgenommen.

Die Präprozessoranweisung *#line* ist im DRC860 *nicht* implementiert.
Die Datei CTEMP.TOK kann mit Hilfe des Programms DRCRPP in eine lesbare Darstellung überführt werden, so daß die Richtigkeit des Textersatzes geprüft werden kann.
Der C-Codegenerator verarbeitet die Datei CTEMP.TOK und erzeugt die Objektdatei. Mit Hilfe des Disassemblers DRC862.CMD ist es möglich, ein Listing zu schaffen, das die C-Anweisungen und deren Umsetzung in Assemblersprache des 8086 enthält. Treten während der Verarbeitung des Quellprogramms Fehler auf, werden diese mit Nummern spezifiziert. Mit Hilfe der Datei DRC.ERR erfolgt eine Interpretation der Fehlerursache.

Die verschiedenen Subprogramme werden vom Programm DRC.CMD aufgerufen. Dabei können verschiedene Optionen für die Spezifizierung der Abarbeitung und das zu erzeugende Programm gesetzt werden.

10.1.2. Aufruf des C-Compilers

Zum Start des C-Compilers wird dieser mit dem Namen DRC aufgerufen:

DRC [d:]cprog [–option . . .]

Dabei ist *cprog* der Name der zu übersetzenden Quelldatei. Wird die Datei *cprog* nicht gefunden, wird die Datei *cprog.c* auf dem spezifizierten Laufwerk gesucht. Mit Hilfe von Optionen kann der Übersetzungsvorgang gesteuert und die Abarbeitung der Subprogramme beeinflußt werden.

Beispiel:
A>DRC TEST.C

Der Compiler sucht die Datei TEST.C auf Laufwerk A, und wenn sie vorhanden ist, wird mit der Übersetzung begonnen.

Beispiel:
A>DRC B:TEST

Jetzt sucht der C-Compiler die Datei TEST auf Laufwerk B. Falls er sie nicht findet, sucht er anschließend die Datei TEST.C. Falls keine der beiden Dateien vorhanden ist, folgt eine Fehlerausschrift.
Durch Betätigen einer beliebigen Taste kann die Arbeit des C-Compilers unterbrochen werden. Es erfolgt dann die Ausschrift

STOP DRC (Y/N)?

und mit der entsprechenden Eingabe wird der Compilerlauf abgebrochen.

10.1.3. Optionen des C-Compilers

Zusätzlich zur Angabe des Quellennamens können verschiedene Optionen, die die Arbeitsweise des C-Compilers beeinflussen, angegeben werden. Diese Optionen können in beliebiger Reihenfolge auftreten und beginnen stets mit einem "–". Bei Angabe eines wahlweisen Parameters ist darauf zu achten, daß zwischen Option und folgendem String kein Leerzeichen stehen darf.
Einige der wichtigsten Optionen sollen hier in alphabetischer Reihenfolge vorgestellt und erläutert werden. Anschließend werden dann einige Beispiele für die Anwendung und die Wirkung von Optionen gezeigt.
–a[string]
ist eine Steueroption für das Programm DRC.CMD und bewirkt das automatische Linken des geschaffenen Programms im Anschluß an die Übersetzung mit Hilfe des Linkers LINK86. Dabei ist *string* die Parameterzeile für LINK86 gemäß 6.4.2. Wird kein *string* angegeben, wird nur das vom C-Compiler in Bearbeitung befindliche Programm gebunden.
–b
Auswahl des Übersetzungsmodells
Bei Setzen dieser Option wird ein Programm für das Large-Modell geschaffen.
–f
bewirkt die Generierung von 8087-Befehlen und ermöglicht so die Einbeziehung des Arithmetikprozessors.

–id:
spezifiziert das Laufwerk **d:**, von dem die Datei <name> der #include-Anweisung des Präprozessors geladen wird.
–L[d:]name
Schaffung eines Listings, das die Include-Dateien auflistet und die Verschachtelungstiefe von Blöcken anzeigt. Wird kein *name* angegeben, erfolgt die Ausgabe der Listdatei auf dem Bildschirm.
–o[d:]name
spezifiziert das Laufwerk und den Namen des vom Codegenerator geschaffenen Objektfiles. Ist diese Option nicht gesetzt, wird der Name des Quellfiles mit dem Nachsatz .OBJ versehen.
–p
beschränkt die Abarbeitung auf den Präprozessordurchlauf.
–r[d:][name]
Es wird der Reassembler DRC862.CMD aufgerufen, und auf Laufwerk *d:* wird die Datei *name* generiert, die das C-Programm und die entsprechenden Maschinenbefehle enthält. Ist kein Dateiname angegeben, erfolgt die Ausgabe auf dem Bildschirm.
–snumber
beschränkt die Zahl der signifikanten Zeichen jedes Symbols auf *number.* Wird die Option *–s* nicht gesetzt, sind die ersten 40 Zeichen eines Symbols signifikant. Fehlt die Angabe *number,* entstehen Fehler beim Präprozessorlauf, da dann Symbollänge Null angenommen wird und die verschiedenen Symbole nicht unterschieden werden können.
–0d:
der Präprozessor wird nicht vom aktuellen Laufwerk, sondern vom Laufwerk **d:** aufgerufen.
–1d:
R.CMD und *DRC861.CMD* werden von Laufwerk **d:** aufgerufen.
–2d:
Starten des Reassemblers *DRC862* von Laufwerk **d:**
–3d:
Starten des Linkers *LINK86* von Laufwerk **d:** (nur bei gesetzter –a-Option).
–4d:
Laden der Fehlerinterpretationsdatei *DRC.ERR* von Laufwerk **d:**
–?
Diese Option zeigt während des Compilerlaufes die gesetzten Optionen an.

Beispiele:
A>DRC test –s8

Von allen Symbolen werden nur die ersten acht Zeichen ausgewertet. Damit ist der übliche Standard eingestellt.

A>DRC test –a –b

Dieses Kommando wird als
A>DRC test
A>LINK86 test
interpretiert. Die Codegenerierung erfolgt für das Large-Modell.

A>DRC e:test –ab:prog=test1, e: test, test2

Nun kommt die Folge
A>DRC e:test
A>LINK86 b:prog=test1 e:test, test2
zur Ausführung.

E>A:DRC test –ia: –a –0a: –1a: –3b: –4a:

Es wird die auf dem aktuellen Laufwerk E befindliche Datei *test* (bzw. *test.c*) übersetzt. Dabei werden Präprozessor, der Codegenerator und die Fehlerdatei auf Laufwerk *A* gesucht. Nach der Generierung des Objektfiles wird auf dem Laufwerk *B* das Programm LINK86 gestartet. Man erhält ein Programm *test.cmd,* das durch Aufruf von *test* ausgeführt werden kann. Objektdatei, CMD-Datei und alle temporären Dateien werden auf Laufwerk *E* angelegt.

A>DRC test –re:test.lst

Alle benötigten Programme befinden sich diesmal auf dem Laufwerk *A.* Als Ergebnis dieses Programmaufrufs entsteht auf dem Laufwerk *E* ein Reassemblerlisting mit dem Namen TEST.LST.

10.1.4. Linken des Programms
Nachdem alle Quellmodule eines Programms übersetzt wurden, müssen sie gebunden (gelinkt) werden, um eine ausführbare Programmdatei zu erhalten. Das ist aus verschiedenen Gründen notwendig. Erstens sind die vom Codegenerator erzeugten Objektdateien nicht ausführbar, und zweitens benutzen die meisten Programme Funktionen, die nicht in der gleichen Datei vereinbart wurden. Solche externen Funktionen können durch den Nutzer in anderen Programmen vereinbart werden und separat zu den Objektdateien übersetzt worden sein oder sich in Programmbibliotheken befinden. Diese Bibliotheken, die die auf den meisten Systemen vorhandenen Standardfunktionen (z.B. *printf(), scanf(), fgetc()...)* enthalten, sind im Falle des DRC-Compilers die Dateien CLEARS.L86 und CLEARL.L86. Je nach dem bei der Übersetzung gewählten Speichermodell werden die benötigten Funktionen vom Linker selbständig in der entsprechenden Bibliothek gesucht und in die ausführbare Programmdatei eingebunden. Die Bibliotheksdatei muß sich dabei auf dem aktuellen Laufwerk befinden.
Obwohl die Programmiersprache **C** eine Funktion *main()* als Startpunkt eines C-Programms definiert, sind eine Reihe von systemabhängigen Programmen notwendig, bevor *main()* aufgerufen wird. Solche Programme fragen beispielsweise die Versionsnummer des Betriebssystems und die Verfügbarkeit eines Arithmetikprozessors ab. Außerdem stellen sie die Argumente *argc* und *argv* für das Hauptprogramm bereit. Auch diese Funktionen sind in den Bibliotheken enthalten. Bei den Argumenten der Hauptfunktion gibt es noch eine Besonderheit: Normalerweise zeigt der Pointer **argv[0]* auf den Namen des laufenden Programms. Da dieser Name unter CP/M-86 zur Laufzeit aber nicht verfügbar ist, enthält *argv[0]* einen Zeiger auf den Namen *Cprogram.*
Das Binden der Programme wird durch den LINK86 realisiert, der entweder im Anschluß an den Compilerlauf separat gestartet wird oder bereits vom C-Compiler aufgerufen wird (–a-Option). Dabei müssen sich die Bibliotheksdateien auf dem aktuellen Laufwerk befinden. Sie brauchen jedoch nicht in der Kommandozeile für den Linker mit angegeben werden, sondern die entsprechenden Module werden automatisch eingebunden.

10.1.5. Symbolisches Debugging
Die Debugger *DDT86* und *SID86* wurden bereits in Abschnitt 6.5. ausführlich beschrieben (siehe auch /4/). Für die Testung und Einzelschrittabarbeitung von C-Programmen ist es günstig, den symbolischen Debugger SID86 zu nutzen, da die beschriebenen Befehle mit Namen und Symbolen spezifiziert werden können. In der Symboltabelle sind globale Variablen und alle vom Nutzer definierten Funktionsnamen enthalten, die nicht „static", das heißt nur im Übersetzungsmodul bekannt sind. Statische Funktionsnamen werden durch den Namen der Quelldatei spezifiziert, versehen mit der Nummer ihres Auftretens in dieser Datei.

Beispiel:
g,.c.c2

Mit diesem Kommando wird das zweite Unterprogramm der Quelldatei *c.c* erreicht. Außerdem ist es günstig, wenn vom zu testenden Programm mit Hilfe des Reassemblers (Option –r) ein Listing angefertigt wurde, das die C-Anweisungen mit den dazugehörigen Assemblerbefehlen enthält. Das erleichtert das Verfolgen des Programmablaufes.
Nach dem Aufruf des symbolischen Debuggers und dem Laden der Symboldatei kann das Programm im Einzelschrittbetrieb abgearbeitet werden. Adressen und Werte, die in der Symboltabelle enthalten sind, können dabei durch *.name* angegeben werden.
Der Instructionpointer IP steht auf der Startadresse. Die hier stehenden Instruktionen haben aber keine Ähnlichkeit mit den Befehlen des Disassemblerlistings. Es handelt sich hier um die in Abschnitt 10.1.4. erwähnten Funktionen, die vor Starten des Programms *main()* ausgeführt werden müssen.

Beispiel:
g,.MAIN

Mit Hilfe dieses Kommandos gelangt man nun an die Stelle, wo die eigentliche Abarbeitung des C-Programms beginnt. Der Datenbereich, der mit der Adresse *name* beginnt, kann durch das Kommando
d.name
im aktuellen Datensegment angezeigt werden. Es ist auch möglich, Variablennamen mit Registern zu indizieren.

Beispiel:
des:.x+bp

Es wird der Bereich im Extrasegment angezeigt, der mit der Adresse x, vergrößert um den Inhalt von bp, beginnt.
So ist ein recht komfortables Testen des Programms möglich. Allerdings ist dabei die Kenntnis der Assemblersprache des Prozessors Voraussetzung. Wenn diese Bedingung nicht erfüllt ist, muß das Testen des Programms durch zweckmäßiges Einfügen von Ausgabeanweisungen mit den jeweils interessierenden Daten erfolgen. Das ist jedoch recht aufwendig, weil zwischen zwei Tests der vollständige Entwicklungsweg (Editieren, Compilieren, Linken) notwendig ist.

10.2. Systemabhängigkeit des C-Programms

Die Programmiersprache **C** ist auf vielen verschiedenen Systemen implementiert worden. Daraus ergeben sich unterschiedliche Varianten der Umsetzung der Hochsprache in die entsprechende Maschinensprache. Für verschiedene spezielle Anwendungsfälle ist es notwendig zu wissen, wie das übersetzte C-Programm aussieht.

10.2.1. Datenelemente

Die Tafel 10.1 gibt eine Übersicht über die Implementierung der verschiedenen Datentypen. Es ist zu beachten, daß die Bezeichnungen *"short"*, *"long"* und *"unsigned"* nur auf den Datentyp *int* angewendet werden können. Das bedeutet beispielsweise, daß der Datentyp *unsigned long* nicht implementiert ist. Außerdem ist, abweichend vom Standard, *char als unsigned* implementiert. Den Datentyp *enum* gibt es nicht. Der Datentyp *void* wird aus Kompatibilitätsgründen in der Header-Datei *PORTAB.H* durch einen Kommentar definiert. *float* definiert eine 32 Bit lange, vorzeichenbehaftete Gleitpunktzahl mit einem Exponenten der Länge 8 Bit und einer Mantisse von 24 Bit Länge in Hidden-Bit-Darstellung. Der Offset des Exponenten beträgt 127. Dieser Datentyp realisiert eine Genauigkeit von etwa 6 bis 7 Dezimalstellen. Der Datentyp *double* definiert eine 64 Bit lange vorzeichenbehaftete Gleitpunktzahl mit 11 Bit Exponent und 53 Bit Mantisse in Hidden-Bit-Darstellung. Das bedeutet eine Genauigkeit von 15 bis 16 Dezimalstellen. Diese interne Darstellung ist äquivalent der vom 8087-Arithmetikprozessor verwendeten Speicherdarstellung von Gleitpunktzahlen.

Tafel 10.1. Datentypen des DRC-C-Compilers

Typ	Bits	Bereich
char	8	0 bis 255
int	16	−32768 bis +32767
short	16	−32768 bis +32767
unsigned	16	0 bis 65535
long	32	$-2 \cdot 10^9$ bis $2 \cdot 10^9$
float	32	$\pm 10^{-37}$ bis $\pm 10^{37}$
double	64	$\pm 10^{-307}$ bis $\pm 10^{307}$

Wenn *Float*-Variablen in C-Programmen verwendet und an Unterprogramme übergeben werden, erfolgt vor der entsprechenden Berechnung eine Umwandlung in den Datentyp *double*, das heißt, die Berechnung erfolgt immer mit höherer Genauigkeit, und erst bei der Abspeicherung des Ergebnisses wird auf die ungenauere Darstellung gerundet.
Pointer auf verschiedene Datentypen sind je nach gewähltem Speichermodell 2 oder 4 Byte lang.
Es können beliebig viele Daten vom Typ *integer* als *register* vereinbart werden. Davon wird die erste im Register SI und die zweite in DI gespeichert. Alle weiteren Registervereinbarungen werden wie *Auto*-Variablen behandelt und entsprechend im Stack abgelegt.

10.2.2. Speicheradressierungsmodelle

Die segmentierte Architektur des 8086 bringt spezielle Probleme für die Implementierung einer Hochsprache mit sich. Um dem Programmierer die Möglichkeit zu geben, eine Kombination von Adressierbarkeit und Effizienz zu wählen, unterstützt der DRC-Compiler zwei Speicheradressierungsarten (auch Speichermodelle genannt).

- Das Small-Modell

Ein kleines Programm, das nur einen geringen Speicherbedarf hat, kann innerhalb eines einzigen Segments plaziert werden. Wenn auch die Menge der zu bearbeitenden Daten die Größe von 64 KByte nicht übersteigt (einschließlich Stack), sind zum Aufruf von Funktionen bzw. für die Adressierung von Speicherplätzen nur die Offsets der vollständigen Adressen erforderlich. Für einen Pointer wird ein 16-Bit-Wort reserviert. Die –b-Option ist in diesem Falle *nicht* gesetzt.
Der Compiler schafft nur zwei Segmente mit je bis zu 64 KByte Länge: **CODE** und *DATA*. Vor Beginn der Abarbeitung des Hauptprogramms wird das Codesegmentregister CS mit der Segmentadresse geladen, und den Segmentregistern DS, ES und SS wird **DATA** zugewiesen. Diese Werte bleiben dann während der gesamten Laufzeit des Programms unverändert. Der Speicher wird für das Small-Modell gemäß Bild 10.1 aufgeteilt.

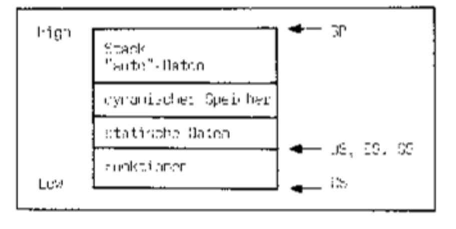


Bild 10.1 Speicheraufteilung im Small-Modell

- Das Large-Modell

Ist das Programm umfangreicher bzw. sind größere Datenmengen zu verarbeiten, ist es notwendig, die absolute Speicheradressierung zu verwenden, das heißt, für jeden Speicherzugriff und Funktionsaufruf muß die vollständige Adresse, bestehend aus Segment und Offset, angegeben werden. Zu diesem Zwecke gibt es das Large-Modell, das durch Setzen der –b-Option ausgewählt wird. Statische und dynamische Daten werden in verschiedenen Segmenten abgelegt. Ein Pointer ist in dieser Adressierungsart 4 Byte lang. Die Aufteilung des Speichers erfolgt nach Bild 10.2.

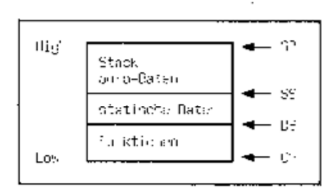


Bild 10.2 Speicheraufteilung im Large-Modell

10.2.3. Funktionsrealisierung

In diesem Abschnitt wird dargestellt, wie Funktionsaufrufe vom C-Compiler realisiert werden.
Wenn ein C-Programm eine Funktion aufruft, werden zunächst die Werte der Argumente auf dem Stack abgelegt, und dann wird der Funktionsruf (CALL) ausgeführt. Das Abkellern der Argumente erfolgt in umgekehrter Reihenfolge (von rechts nach links), so daß sie der aufgerufenen Funktion in der natürlichen Reihenfolge zur Verfügung stehen (von links nach rechts mit steigenden Adressen).

Beispiel:
Der Aufruf der Funktion *func(x,y,z)* mit den drei Integerargumenten x, y und z erzeugt eine Stackaufteilung nach Bild 10.3.
Von der gerufenen Funktion werden dann die folgenden Aktionen ausgeführt:
① Das BP-Register wird auf dem Stack abgelegt und so der Wert des aufrufenden Programms gesichert.
② Das Basisregister BP wird mit dem Inhalt des Stackpointers SP geladen, um eine Adressierung der automatischen und temporären Variablen auf dem Stack mit Hilfe von BP zu ermöglichen.
③ Verwendet die aufgerufene Funktion *Register*-Variablen, werden die Register SI und DI in den Stack gespeichert.
④ Der Stackpointer SP wird um *die* Bytezahl verringert, die von den automatischen Variablen der aufgerufenen Funktion benötigt werden. Dieser Stackbereich umfaßt alle *Auto*-Daten, die in der Funktion deklariert wurden und kann auch einen zusätzlichen Bereich für temporäre Speicherplätze, die oftmals bei der Lösung von Gleichungen benötigt werden, enthalten. Wenn keinerlei automatische oder temporäre Variablen von der aufgerufenen Funktion verwendet werden, wird dieser Schritt übersprungen.
Der Stack hat dann die Anordnung nach Bild 10.4.
Die Adressierung der Argumente und der automatischen Daten erfolgt immer relativ zu BP.
Bevor eine Funktion zur aufrufenden Funktion zurückkehrt, wird der Rückgabewert (falls vorhanden) in bestimmte vorgegebene Register geladen. Welche Register dabei benutzt werden, wird durch das Format des Rückgabewertes entsprechend der folgenden Tafel bestimmt.

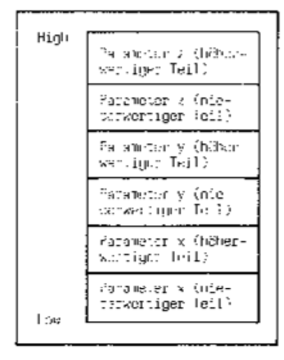


Bild 10.3 Anordnung der Funktionsparameter im Prozessorstack

Länge des Rückgabewertes	Register	Beispiel
8 Bit	**AL**	char
16 Bit	**AX**	int, Pointer im Small-Modell
32 Bit	**(BX, AX)**	long, float, Pointer im Large-Modell
64 Bit	**(DX, CX, BX, AX)**	double

Wenn mehrere Register verwendet werden, enthält *AX* den *niederwertigsten* Teil des Ergebnisses. Das heißt, wenn im Large-Modell ein Pointer zurückgegeben wird, enthält BX die Segmentadresse und AX die Offsetadresse.
Aus der Organisation der Übergabe des Funktionswertes geht hervor, daß *keine Strukturen* an die aufrufende Funktion zurückgegeben werden können. Als *Argument von Funktionen* sind Strukturen dagegen zugelassen. Sie werden vollständig in den Stack übertragen.
Wenn die Funktion beendet wird, erfolgt zunächst die Rücksetzung des SP um den Bereich der automatischen Variablen und nach dem Rückretten eventuell gekellerter SI- und DI-Register und des Basepointers BP erfolgt

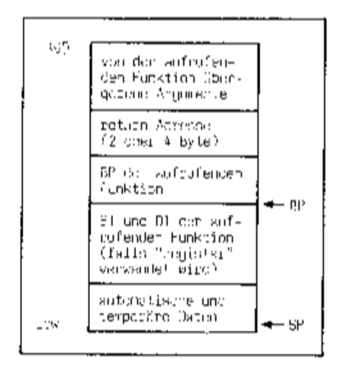


Bild 10.4 Inhalt des Stack bei Ausführung der Funktion

je nach Speichermodell ein NEAR- oder FAR-Return. Weil eine C-Funktion in besonderen Fällen auch eine unterschiedliche Zahl von Argumenten haben kann (z. B. printf()), kann die aufgerufene Funktion den von den Parametern belegten Speicherplatz nicht freigeben (d. h. der Befehl *ret n* wird nicht verwendet), sondern dieses Rücksetzen des SP wird von der aufrufenden Funktion durchgeführt.

10.2.4. Einbindung maschinen- bzw. betriebssystemabhängiger Programmanteile in C-Programme

In einigen Fällen der systemnahen Programmierung erweisen sich die in den Laufzeitbibliotheken enthaltenen Standardfunktionen als nicht ausreichend. Das ist speziell dann der Fall, wenn maschinenspezifische Programme abgearbeitet werden sollen, wie es beispielsweise bei Portein-/Portausgaben und beim Einbinden spezieller 8087-Programme erforderlich ist.

- Verbindung von Assembler- und C-Programmen

Es ist möglich, Assemblermodule für die Einbindung in C-Programme zu schreiben, da der Assembler RASM86 ein gleich aufgebautes Objektfile liefert (siehe Abschnitt 6) und der Linker LINK86 mehrere solcher Objektdateien verbinden kann. Allerdings sind beim Erzeugen solcher Assemblerdateien verschiedene Besonderheiten zu beachten:
① Der Name des von einer C-Funktion aufzurufenden Assemblerprogramms muß in der A86-Datei als PUBLIC vereinbart werden, damit er dem Linker bekannt ist.
② Da der Assembler alle Buchstaben als Großbuchstaben behandelt, muß der Name der Funktion auch im C-Programm mit großen Buchstaben geschrieben werden.
③ Die Auswertung der vom C-Programm übernommenen Parameter muß den unter 10.2.3. beschriebenen Konventionen für die verschiedenen Speichermodelle entsprechen. Das bedeutet insbesondere, daß Funktionen, die mit CALLF aufgerufen wurden (Large-Modell), mit RETF abgeschlossen werden. Auch ist der unterschiedliche Offset der übergebenen Parameter relativ zu BP – bedingt durch die unterschiedliche Länge der Return-Adresse – bei den verschiedenen Speichermodellen zu beachten.
④ Wenn ein Funktionswert zurückgegeben werden soll, muß er in die entsprechenden Register (siehe Punkt 10.2.3.) eingetragen werden.

```
            CSEG
INP:        PUSH BP
            MOV BP,SP
            MOV DX,X[BP]
            IN AL,DX
            XOR AH,AH
            POP BP
;
IF MODEL
            RETF
ELSE
            RET
ENDIF
```

- Einbinden von CP/M-BDOS-CALLs

Viele der implementierten Standardfunktionen nutzen die BDOS-CALLs des Betriebssystems (z. B. Tastaturein-/Tastaturausgabe). Es ist aber auch möglich, in Anwenderprogrammen BDOS-CALLs zu nutzen, falls die Standardfunktionen nicht ausreichen. Das geschieht mit Hilfe der Funktion *_BDOS(nr,value)*.
Der Parameter *nr* wird in das Register CX geladen und spezifiziert die Nummer des BDOS-Calls. Der Wert *value* wird in das Register DX übernommen.
Die Funktion *__BDOS()* hat folgende Wirkung: Die externe Variable *_cpmrvcx* enthält den vom BDOS-CALL in CX zurückgegebenen Wert, und in *_cpmrv* wird der Wert des Registers AX geladen. Dieser Wert ist natürlich auch als Funktionswert der Funktion verfügbar. Das folgende Beispiel zeigt die Einbindung der Abfrage des Tastaturstatus in ein C-Programm mit Hilfe von BDOS-CALLs.

Beispiel:

```
#include <std.h>
main()
{
   char c;
   ...
   if(__BDOS(11)){
   /* Abfrage des Tastaturstatus */
   /* Taste gedrückt */
      ...
   } else {
   /* Taste nicht gedrückt */
      ...
   }
   ...
}
```


Literatur

/1/ Dokumentation für Echtzeitbetriebssystem BOS 1810. VEB Kombinat Robotron
/2/ Claßen, L.; Oefler, U.: UNIX und C. VEB Verlag Technik Berlin: 1987
/3/ Horn, T.: Programmierung in C. Mikroprozessortechnik, Berlin 1(1987) 1–6
/4/ Kernighan, B. W.; Ritchie, D. M.: Programmieren in C. München, Wien: Carl-Hanser-Verlag, 1983
/5/ Dokumentation SCP 1700, VEB Robotron Elektronik Dresden

Schluß

TERMINE

VIDEOTON-Ausstellung

WER? VIDEOTON
WANN? 24. bis 28. April 1989
WO? Botschaft der Ungarischen Volksrepublik in Berlin
WAS?
- Rechentechnik
- Robotertechnik
- Fernsehgeräte
- Nachrichtentechnik

WIE? Ausführliche Informationen über das Programm erhalten Sie in den VIDEOTON-Vertretungen der DDR: Handelsvertretung der Botschaft der UVR, Tel. 2202561, Kundendienstbüro Berlin, Tel. 4724185/86, Kundendienstbüro Erfurt, Tel. 711472.

Görs

Hinweis

Im Beitrag „Indizierte Variablen und Felder unter REDABAS" von Dr. Thomas Streubel in MP 3/89, S. 89, muß es in der zweiten Spalte in den Zeilen 4 und 5 richtig heißen:
store substr(str((&ind+1001),4,0),2,3) to ind
(16 Bit)
Autor und Redaktion bitten, den Fehler zu entschuldigen.

Environment-Support für Turbo-Pascal

Christian Hanisch, Berlin

Komplexere Turbo-Pascal-Anwendungen bestehen meist aus mehreren Komponenten (Overlays, Konfigurationsdateien, CHN-Files, etc.), die durch das Hauptprogramm zur Laufzeit gegebenenfalls nachgeladen werden müssen. Um nun zu gewährleisten, daß zur Abarbeitung nicht unbedingt in das Directory der besagten Turbo-Pascal-Anwendung gewechselt werden muß, sind eine Reihe von Varianten denkbar. Angefangen bei Batch-Files über fest vorgeschriebene Pfade für die Anwendung bis hin zur Installation der Software in ein nutzergewähltes Directory mittels spezieller Installationsprogramme, existieren hier Möglichkeiten mit ganz unterschiedlichen Anforderungen an die programmtechnische Realisierung. Eine völlig unproblematische Lösung für den Anwender stellt jedoch nur die Variante des automatischen Findens aller zu einem System gehörenden Dateien dar. Voraussetzung zum Praktizieren einer solchen Lösung ist, daß sich Hauptprogramm und zugehörige Systembestandteile in einem Directory befinden. Dann bieten sich unter DOS zwei Wege an:

1. Das Hauptprogramm wertet eine spezielle mit dem **SET**-Befehl in das Environment vorgenommene Eintragung, die den Zugriffspfad auf die Anwendung beinhaltet (z. B. **SET VSLCOLOR=C:\SYS\CL\VSL**) aus und findet so die Dateien des Softwaresystems.
2. Das Hauptprogramm bestimmt zur Laufzeit den Pfad, unter dem es selbst und alle zugehörigen Bestandteile zu finden sind.

Hier soll nun die Realisierung der 2. Variante in Turbo-Pascal (Version 3.xx und 4.0) gezeigt werden.

Unter DOS gehört zu jedem im RAM ablaufenden Programm (Process) ein Environment-Block der die mit **PATH, PROMPT, COMSPEC** und **SET** eingetragenen Informationen beinhaltet. Ein laufendes Programm kann damit durch Auswertung der Environment-Einträge z. B. den mit **PATH** gesetzten Zugriffspfad auf Kommandodateien nutzen.

Ab DOS 3.0 befindet sich hinter den Environment-Einträgen (Ende-Kennzeichnung: 00 00) eine Kopie der Dateibezeichnung des aufgerufenen Programms mit Laufwerk und Zugriffspfad (siehe Bild 1).

Die im Bild 2 dargestellte Funktion Get_Calling_Path nutzt nun die Tatsache, daß im PSP (Program-Segment-Prefix) ab 2CH ein Zeiger auf das Environment des jeweiligen Prozesses steht, zur Bestimmung des Beginns des Environments. Danach wird das Ende des Environments (00 00) gesucht, der Zeiger dorthin gestellt und ab Offset 04H die Laufwerksbezeichnung und der Zugriffspfad bis zur Dateibezeichnung des laufenden Programms kopiert. Die Dateibezeichnung des laufenden Programms muß der Funktion daher beim Aufruf übergeben werden. Der von der Funktion gelieferte String kann nun benutzt werden, um Pfadnamen für nachzuladende oder zu lesende [Konfigurations-] Dateien zu bilden oder mittels OvrPath den Overlay-Pfad unter Turbo-Pascal Version 3.xx zu setzen. Durch eine kleine Modifikation der Funktion ist es auch möglich, das Environment nach bestimmten Einträgen abzusuchen, so daß der oben erwähnte erste Weg praktikabel wird.

```
-D ES:2C L 4  <- PSP von ENVIRON.EXE anzeigen      ┌── Zeiger auf Beginn Env.
                                                   │   im PSP von ENVIRON.EXE
1B38:0020                                      [28 1B] 4C 01                (.L.
-D 1B28:0 L F0

1B28:0000  43 4F 4D 53 50 45 43 3D-43 3A 5C 43 4F 4D 4D 41   COMSPEC=C:\COMMA
1B28:0010  4E 44 2E 43 4F 4D 00 50-52 4F 4D 50 54 3D 24 70   ND.COM.PROMPT=$p
1B28:0020  24 67 00 4C 49 42 3D 63-3A 5C 73 79 73 5C 63 6C   $g.LIB=c:\sys\cl
1B28:0030  5C 66 37 37 5C 4C 49 42-00 54 4D 50 3D 63 3A 5C   \f77\LIB.TMP=c:\
1B28:0040  73 79 73 5C 63 6C 5C 66-37 37 5C 54 4D 50 00 50   sys\cl\f77\TMP.P
1B28:0050  41 54 48 3D 43 3A 5C 53-59 53 32 5C 50 52 4F 43   ATH=C:\SYS2\PROC
1B28:0060  4C 49 42 3B 43 3A 5C 53-59 53 32 5C 43 4C 5C 45   LIB;C:\SYS2\CL\E
1B28:0070  58 45 3B 43 3A 5C 3B 43-3A 5C 53 59 53 5C 50 52   XE;C:\;C:\SYS\PR
1B28:0080  4F 43 4C 49 42 3B 43 3A-5C 53 59 53 5C 43 4C 5C   OCLIB;C:\SYS\CL\
1B28:0090  46 37 37 5C 42 49 4E 3B-43 3A 5C 53 59 53 5C 43   F77\BIN;C:\SYS\C
1B28:00A0  4C 5C 44 4F 53 3B 43 3A-5C 53 59 53 5C 43 4C 5C   L\DOS;C:\SYS\CL\
1B28:00B0  54 53 52 3B 43 3A 5C 53-59 53 5C 43 4C 5C 50 43   TSR;C:\SYS\CL\PC
                                 ┌── Ende des Environment
                               ──┴──
1B28:00C0  54 4F 4F 4C 53 3B 00 00-01 00 43 3A 5C 53 59 53   TOOLS;....C:\SYS
1B28:00D0  32 5C 43 4C 5C 45 58 45-5C 45 4E 56 49 52 4F 4E   2\CL\EXE\ENVIRON
1B28:00E0  2E 45 58 45 00 01 00 8B-E5 5D C3 55 8B EC 83 EC   .EXE.....].U....
-Q
```

Bild 1 Blick in das Environment von ENVIRON.EXE

Bild 2 Funktion Get Calling Path mit Demonstrationsprogramm

```
PROGRAM Environment_DEMO; { TURBO PASCAL Ver. 4.0                       }
{ Das Demo-Program ist als EXE-File zu compilieren und
  unter dem Namen ENVIRON.EXE abzulegen!
  Unter TURBO PASCAL Ver. 3.xx als COM-File ablegen.                    }

USES CRT;    { entfaellt unter Ver. 3.xx                                }

TYPE string80 = STRING[80];
VAR  x         : string80;

{ ------------------------------------------------------------------- }

FUNCTION Get_calling_path(s:string80):string80;
TYPE charstr     = ARRAY[1..162] OF CHAR;
VAR  envptr      : ^charstr;
     environment: ^integer;
     i           : INTEGER;
BEGIN
  FOR i:=1 TO length(s) DO s[i]:=upcase(s[i]);

  { Pointer auf den Beginn des Environments stellen                    }
  environment:=ptr(prefixseg,$2C);
  { Ver. 3.xx: environment:=ptr(cseg,$2c);                             }
  envptr:=ptr(environment^,0);

  { Ende des Environments suchen: Kennzeichen ist 00 00
    danach folgt 01 00 <drive:\path\filename.ext>                      }
  REPEAT
    i:=pos(#0#0,envptr^);
    { Wenn i=0, Pointer zum Fortsetzen der Suche
      weiterruecken, sonst Pointer auf das zweite Byte 00
      des Environment-Ende-Kennzeichens setzen.                        }
    IF i=0 THEN envptr:=ptr(environment^,
                         ofs(envptr^)+length(envptr^)-1)
           ELSE envptr:=ptr(environment^,ofs(envptr^)+i)
  UNTIL i<>0;

  { ab 4. Position Laufwerksbezeichnung und Zugriffspfad
    bis zum Beginn der Dateibezeichnung kopieren und
    fertig!                                                            }
  i:=pos(s,envptr^);
  IF i=0 THEN get_calling_path:='+++ERROR+++'
         ELSE get_calling_path:=copy(envptr^,4,i-4)
END;

{ ------------------------------------------------------------------- }

BEGIN
  clrscr;
  x:=get_calling_path('ENVIRON.EXE');
  { Ver. 3.xx: x:=get_calling_path('ENVIRON.COM');                     }
  gotoxy(40-(length(x) SHR 1),13);write(x);
  { Falls im Memory-Modus getestet wird, musz ENVIRON.EXE
    (.COM) durch TURBO.EXE (.COM) bzw. den Namen der
    TURBO PASCAL-Entwicklungsumgebung ersetzt werden.
    Anwendungsbsp.: Setzen des Overlay-Pfades unter TURBO
                    PASCAL Ver 3.xx mit OVRPATH(X);
                    Bilden des Dateinamens einer Konfigu-
                    rationsdatei filename:=x+'DEMO.CFG';               }
END.
```

Fortschritte in dBase III Plus

Dr. Wilfried Grafik, Humboldt-Universität zu Berlin, Sektion Mathematik

Dieser Beitrag untersucht die Weiterentwicklungen von dBase III Plus gegenüber dBase III. dBase III Plus wurde nur wenig später als dBase III veröffentlicht. Es stellt eine Reaktion auf die Erfahrungen bei der praktischen Arbeit mit dBase III auf 16-Bit-Rechnern dar. Für eine grundlegend neue Version, wie sie jetzt mit dBase IV vorliegt, war die Zeit noch nicht reif. Geräte- und softwaretechnische Entwicklungen und Erfahrungen mußten akkumuliert werden, um einen echten Qualitätssprung zu erreichen. Deshalb bestanden die Ziele von dBase III Plus bei voller Abwärtskompatibilität zu dBase III in Verbesserungen bezüglich folgender Komplexe:
- Nutzung der Hardware
- schnellere Algorithmen
- Service für den Programmierer
- Möglichkeiten der Oberflächengestaltung
- Arbeit mit komplexen Datenbanken
- Kooperation mit anderen Programmen
- Beseitigung einiger Schwächen von dBase III
- Unterstützung eines Mehrnutzerbetriebs.

Eine an der Syntax orientierte Zusammenstellung der Veränderungen von dBase III Plus gegenüber dBase III befindet sich auf der dritten Umschlagseite dieses Hefts.

Bessere Nutzung der Hardware

Die unter MS-DOS von dBase anzusteuernde Hardware ist der Bildschirm, die Tastatur und der Drucker. Die Dateiverwaltung kann, bis auf die Benutzung von Katalogen (Unterverzeichnissen, subdirectories), nicht beeinflußt werden. Der Anschluß eines Plotters ist nicht vorgesehen. dBase III Plus unterstützt keine grafischen Darstellungen, schafft dafür aber bessere Verbindungen zu Programmen, die grafische Darstellungen gestatten. Zur verbesserten Bildschirmansteuerung wurde **SET COLOR TO** geringfügig erweitert. Als Farbwert ist auf den Positionen <std_display> und <erw_display> zusätzlich der Buchstabe x zugelassen. Er bewirkt, daß Zeichen und Kursor im ausgewählten Bereich nicht sichtbar sind: dBase unterdrückt das Echo von der Tastatur auf dem Bildschirm. Ein möglicher Einsatzfall ist die Anforderung von Paßwörtern. Durch Gleichsetzen von Vordergrund und Hintergrund kann der gleiche Effekt erzielt werden. So bewirkt

SET COLOR TO ,R/R

die Darstellung der Eingabebereiche für @-Kommandos in rot (Vordergrund und Hintergrund). Somit sind der eingegebene Text und der Kursor ebenfalls nicht sichtbar. Der Zusatz <hintergrund> ist nur für solche Bildschirmtypen wirksam, die keine zeichenweise Auswahl des Hintergrunds zulassen. Die Funktion **ISCOLOR()** liefert eine Information über die verwendete Bildschirmansteuerung (Farbgrafik oder monochrom). Für Monochrom-Bildschirme sind andere Bildschirmattribute gültig: *U* für unterstreichen und *I* für invers. Die Auswahl bestimmter Schriftsätze (gotisch, deutsch, griechisch usw.), wie sie auf modernen Terminals üblich sind, ist nicht möglich. Der erweiterte ASCII-Zeichensatz kann selbstverständlich verwendet werden. Die Beschränkung auf acht Farben in den Kombinationen *normal, intensiv* und *blinken* bleibt erhalten. Der Bildschirminhalt kann auf der Sprachebene von dBase III Plus nicht gelesen oder gerettet werden, wie es in der vom Clipper-Compiler akzeptierten Sprache durch **SAVE SCREEN** möglich ist.

Die Bedienung der Tastatur ist wesentlich erweitert, was die Gestaltung nutzerfreundlicher Oberflächen gestattet. Die Funktion **INKEY()**, in dBase III nicht dokumentiert, aber in allen getesteten Versionen verfügbar, gestattet die Abfrage fast aller Tasten (der Tastatur), insbesondere der Kursorpositionierungstasten einschließlich ihrer gebräuchlichen Kombinationen mit der Ctrl-Taste, z. B. ^Home, ^End usw. Tasten wie CapsLock, NumLock, Del und einige andere Tasten liefern den Kode 0, was der Information *keine Taste gedrückt* entspricht. Des weiteren sind nicht alle Kombinationen mit Shift, Ctrl und Alt abfragbar.

Für die Ansteuerung des Druckers sind keine Weiterentwicklungen zu vermerken. So ist die Umschaltung auf verschiedene Schriftarten, Schriftbreiten, Zeichen- und Zeilenabstände durch Sprachelemente von dBase nicht möglich. Es müssen die von dBase III bekannten Techniken benutzt werden, die auf der vom Nutzer programmierten Ausgabe von Steuerfolgen an den Drucker basieren (Funktion **CHR**).

Schnellere Algorithmen

dBase III Plus hat effektivere Algorithmen als dBase III, was sich besonders beim Indizieren bemerkbar macht. Um eine quantitative Vorstellung zu bekommen, wurden Messungen auf einem XT-Rechner für dBase III Plus und dBase III unter ansonsten identischen Bedingungen durchgeführt. Die Daten und Kommandos sind die gleichen, die bei den in der MP 10/88 /1/ veröffentlichten Messungen benutzt wurden. Tafel 1 beschränkt sich deshalb auf eine Kurzbezeichnung des getesteten Kommandos, auf die Angabe der benötigten Zeit in Sekunden und auf das Verhältnis zum Vergleichswert für dBase III in Prozent. Diese Messungen testen im wesentlichen die Geschwindigkeit des Zugriffs auf Dateien. Das ist sicher heute noch der dominierende Zeitanteil bei der Arbeit eines Datenbanksystems. Je komfortabler die Nutzeroberflächen werden, je mehr Kontrollen (Integrität, Konsistenz, Formatmasken und -funktionen, Gültigkeitsbereiche) durchgeführt werden, desto höher wird der Zeitanteil, der durch Prozessorarbeit, speziell durch die Ansteuerung des Bildschirms, verbraucht wird. Deshalb wurde auch die Zeit gemessen, die für den Aufbau eines Bildschirminhalts benötigt wird. Als Bildschirminhalt wurde das erste Menübild des SET-Kommandos gewählt. Der Bildschirm wird gelöscht, Kopf- und Fußzeile ausgegeben, zwei Rechtecke mit doppelter Umrandung werden gezeichnet und mit Texten gefüllt. In dBase III wurde zum Zeichnen der Rechtecke das Kommando **BOX** aus der Clipper-Umgebung als Prozedur nachgebildet. 43 Sekunden wurden zum Bildaufbau in dBase III benötigt, 53 Sekunden in dBase III Plus. Wird in dBase III Plus das Kommando **@... TO** eingesetzt, so sind es nur noch 12 Sekunden (Clipper: 2 s bei Verwendung des BOX-Kommandos). In dBase III Plus wurde durch die Hinzunahme weiterer Sprachelemente, insbesondere durch weitere Funktionen, und durch eine verbesserte Implementation einiger Sprachelemente ein besseres Laufzeitverhalten erreicht. dBase III Plus bleibt aber trotzdem hinter übersetzten Programmen (Clipper) und selbst hinter der interpretierenden Version von FoxBase unter MS-DOS zurück.

Tafel 1 Ausführungszeiten für dBase III Plus im Vergleich zu dBase III

Kommando	dBase III Plus	
	Zeit in m:ss	relativ zu dBase III in %
COUNT	0:17	59
DELETE FOR	0:19	95
INDEX 1	1:05	21
INDEX 3	1:05	20
LOCATE FOR	0:20	111
PACK	0:59	98
REPLACE ALL	1:08	93
SORT 1	3:43	66
SORT 3	4:55	70
SUM 1	0:27	67
SUM 5	1:23	132

Service für den Programmierer

Die Schnittstelle zum Programmierer wurde in dBase III Plus wesentlich verbessert. Die Menügestaltung des Assistenten und des Berichtsgenerators wurden grundlegend überarbeitet. Die neu hinzugekommenen Kommandos im Direktmodus (**CREATE:MODIFY QUERY:VIEW:SCREEN**) verwenden viele Pull-down- und Pop-up-Menüs und gestatten so im Dialog die schrittweise Formulierung relativ komplizierter Sachverhalte, beispielsweise die Definition der Abhängigkeiten vorhandener Datenbankfiles untereinander (VIEW und CATALOG) und die Formulierung globaler Filterbedingungen (QUERY). Eine zweite Art von Bildschirmmasken kann durch CREATE SCREEN erzeugt werden. Die Definition erfolgt im Direktmodus, wobei ein Wechsel zwischen den definierenden Daten und der aufgebauten Bildschirmmaske möglich ist (Taste F10). Durch den integrierten Maskengenerator ist die Verwendung separater Programme für diese Zwecke nicht mehr erforderlich. Verbessert wurden außerdem die Hilfestellungen hinsichtlich der benutzten Dateien. So ist es in den meisten Kommandos möglich, die Struktur eines ausgewählten Datenbankfiles zu besichtigen und durch Kursorpositionierung das gewünschte Feld für eine Operation auszuwählen (in dBase III nur bei **REPORT**).

Möglichkeiten der Oberflächengestaltung

Für reale Datenbankanwendungen auf 16-Bit-Rechnern gewinnt die Gestaltung einer nutzerfreundlichen Oberfläche immer mehr an Bedeutung. Gefordert sind Menübilder, in denen sich der Anwender durch die Kursorpositionierungstasten bewegen kann, in denen er gegebenenfalls mehrere Kursoren zur

Verfügung hat und in denen er durch das Drücken bestimmter Tasten Auswahlen trifft und Umschaltungen vornimmt. Das alles muß in einer akzeptablen Geschwindigkeit erfolgen, die den Fortgang der Arbeit nicht träge gestaltet. Der Aufbau komfortabler Menüs ist in dBase III Plus noch zu langsam. Der Aufbau der Bilder kann mit dem Auge verfolgt werden, was allgemein nicht akzeptiert wird. Clipper erreicht hier sehr gute Zeiten. Durch die verbesserte Tastaturbedienung (**INKEY(), READKEY(), ONKEY()**) ist die Bewegung im Menü und die Auswahl aus dem Menü vorerst befriedigend gelöst. Die meisten Nutzermenüs sind daher ein Kompromiß zwischen traditionellen, sequentiell aufgebauten Menüs und kleinen Menüs im Direktmodus.

Durch das Kommando **@...TO** kann ein Rechteck auf dem Bildschirm gelöscht und mit einem Rahmen umgeben werden (Achtung: **CLEAR** und **DOUBLE** können nicht gleichzeitig angegeben werden!). Der darunterliegende Inhalt ist verloren. Derartige Gebilde werden als Box bezeichnet. Sie bleiben auf dem Bildschirm erhalten und können nur durch andere Zeichen, unter Umständen durch ein Rechteck aus Leerzeichen, überschrieben werden. Im Gegensatz dazu ist ein Fenster (window) ein rechteckiges Gebilde, das wieder geschlossen werden kann. Dabei wird der darunterliegende Inhalt wieder sichtbar. Das verlangt entweder, daß der vorher vorhandene Bildschirminhalt gerettet wurde oder daß er anderweitig rekonstruierbar ist. Zum Beispiel durch die Programmlogik. Da dBase III Plus kein Kommando zum Retten des Bildschirminhalts oder sogar von Teilen davon enthält, bleibt nur der zweite Weg. Das sind aber meist rechenintensive Operationen, das heißt, die Geschwindigkeit des Prozessors und die Bildschirmansteuerung bestimmen hier die Geschwindigkeit des Datenbanksystems.

In Anwendersystemen wird die private Behandlung von dBase III Plus-Fehlern durch das **ON ERROR**-Kommando unterstützt.

Arbeit mit komplexen Datenbanken

In der Praxis verwendete Datenbanken bestehen meist aus mehreren Datenbankfiles, zu denen mehrere Index-, Format- und Berichtfiles gehören. In dBase III Plus kommen noch Query- und Viewfiles hinzu. Die zehn möglichen Arbeitsbereiche und die maximal 15 gleichzeitig eröffneten Dateien sind keine unerreichbaren Grenzen mehr. Vier Verbesserungen erscheinen hier erwähnenswert. Mehrere Datenbankfiles können über **SET RELATION** verbunden werden. In dBase III ist nur eine Verbindung möglich. In dBase III Plus kann in jedem Arbeitsbereich höchstens eine Verbindung definiert werden. Ein nachfolgendes SET RELATION in diesem Arbeitsbereich hebt die vorher wirksame Verbindung auf. Die Bildung von Ketten ist möglich, so daß maximal neun Verbindungen wirksam sein können. Zyklen in den Verbindungen werden als Fehler zurückgewiesen. Die Verbindungen zwischen den Files können in einem Viewfile gespeichert werden (**CREATE VIEW**). Durch **SET VIEW TO** <.vue_filename> werden sie aktiv.

Der Überblick über alle in einer Datenbank benutzten Dateien kann durch Benutzung eines sogenannten Katalogs verbessert werden. Ein Katalog ist eine Datei (Standardname catalog.cat), die die Struktur eines Datenbankfiles hat (Tafel 2). Durch **SET CATALOG TO** wird der Katalog im Arbeitsbereich 10 in Benutzung genommen. Dieser Arbeitsbereich kann damit von anderen Datenbankfiles nicht benutzt werden. Im Katalog werden zu jeder benutzten Datei zusätzliche Angaben gespeichert, zum Beispiel ein Kommentartext. Dieser Text wird bei Angeboten von Dateinamen in Menüs in einer Box dargestellt und erleichtert so die Orientierung. Eine weitere Orientierung über die ausgeführten Kommandos kann die sogenannte History liefern (**SET HISTORY**). Sie enthält die zuletzt eingegebenen Kommandos, aber im Unterschied zum Alternatefile nicht deren Wirkung. Die History wird durch **DISPLAY HISTORY** sichtbar. Sinnvoll wäre auch eine wiederholte Korrektur und Ausführung von Kommandos aus der History, was gegenwärtig nicht möglich ist.

dBase III enthält bereits das Kommando **SET FILTER TO** <bedingung>. Die angegebene Bedingung wird durch logisches UND mit den Auswahlbedingungen der einzelnen Kommandos verknüpft und schränkt so die für die Operation in Frage kommenden Datensätze ein. dBase III Plus gestattet darüber hinaus die Abspeicherung von Filterbedingungen in Queryfiles (**CREATE QUERY**). Wird ein Queryfile aktiviert (**SET FILTER TO FILE** <.qry_filename>), so werden die dort definierten Filterbedingungen wirksam.

Die Einführung von View- und Queryfiles ist als eine Annäherung von dBase an das relationale Datenbankmodell zu betrachten.

Tafel 2 Struktur des Katalogfiles

Structure for database: C:catalog.cat
Number of data records: 2
Date of last update: 02/06/89

Field	Field Name	Type	Width	Dec
1	PATH	Zeichen	70	
2	FILE NAME	Zeichen	12	
3	ALIAS	Zeichen	8	
4	TYPE	Zeichen	3	
5	TITLE	Zeichen	80	
6	CODE	Numerisch	3	
7	TAG	Zeichen	4	
** Total **			181	

Kooperation mit anderen Programmen

Auf der Ebene des Datenaustauschs können in dBase III Textfiles verarbeitet werden (SDF – standard data format), und es sind Techniken des Zugriffs auf Datenbankfiles aus anderen Programmiersprachen bekannt. dBase III Plus verarbeitet auch Datenformate der Programme VisiCalc (DIF), Multiplan (SYLK) und Lotus 1–2–3 (WKS) in den Kommandos **COPY** und **APPEND**. Die neuen Kommandos **EXPORT** und **IMPORT** gestatten die Verarbeitung von Files in einem Standardformat (PFS), welches ebenfalls von den oben genannten Programmen verarbeitet werden kann.

Zur Kooperation auf der Programmebene stehen neben dem **RUN**-Kommando die Kommandos **LOAD** und **CALL** zur Verfügung. Mit LOAD können sogenannte Binärfiles (.bin) in den Speicher geladen und durch CALL aufgerufen werden. Die Speicherverwaltung ist dabei dem Regime des Betriebssystems vollkommen untergeordnet. In dieser Hinsicht unterscheiden sich diese Kommandos im Niveau wesentlich von dem in dBase II bekannten Kommando CALL. Ein Binärfile ist das Resultat einer Übersetzung aus einer beliebigen Quellsprache und einer anschließenden Überarbeitung mit dem Programm EXE2BIN.

Beseitigung einiger Schwächen von dBase III

Die oben aufgeführten Verbesserungen von dBase III Plus beheben natürlich auch einige Schwächen von dBase III, so etwa die Beschränkung auf eine mögliche Verbindung zwischen Datenbankfiles (SET RELATION). Weitere Kritiken an dBase III bezogen sich auf die Datenerfassung und die Standardberichte (REPORT). Für die Datenerfassung ist der integrierte Maskengenerator (CREATE SCREEN) ein Fortschritt. Die Probleme nutzereigener Menüs im sequentiellen und Direktmodus wurden bereits besprochen. Hinsichtlich der Formatkontrollen, der Einhaltung von Grenzen (**@...SAY....GET ...RANGE**), der Konsistenz und der Widerspruchsfreiheit der Daten, sowie der Überprüfung globaler Restriktionen wurden keine spürbaren Fortschritte in dBase III Plus erreicht. Berichte werden zwar in einem besseren Menü erstellt, sind aber auch nur zweistufig möglich.

Die Arbeit mit dem Typ Datum wurde durch **SET CENTURY ON** verbessert. Das Jahrhundert wird angezeigt und kann mit erfaßt werden, beispielsweise bei **APPEND, EDIT, BROWSE** usw. Datumskonstanten sind nicht verfügbar. Hier muß die Konvertierungsfunktion **CTOD** angewendet werden. dBase III Plus akzeptiert bei der Eingabe Jahreszahlen zwischen 100 und 9999, rechnet aber auch mit Datumswerten darunter. Die Datumsform kann durch **SET DATE** den nationalen Besonderheiten angepaßt werden.

Unterstützung eines Mehrnutzerbetriebs

Eines der ersten Argumente für dBase III Plus ist seine *Netzfähigkeit*. Tatsächlich kann dBase III Plus an einer geeigneten Anlage mit mehreren Terminals, der entsprechenden Netzsoftware und in einer dazu passenden Konfiguration von dBase III Plus im Mehrnutzerbetrieb arbeiten. Wichtig ist dabei, daß die benutzten Installationen von dBase III Plus die entsprechenden Netzkomponenten beinhalten. Wurde eine Installation für den Einnutzerbetrieb erworben, so ist diese in der Regel nicht netzfähig! Funktionstüchtig sind aber die Kommandos von dBase III Plus, die für den parallelen Zugriff mehrerer Nutzer auf die gleiche Datenbank erforderlich sind. So werden **USE...EXCLUSIVE, SET EXCLUSIVE** und **UNLOCK** akzeptiert. **ACCESS()** liefert 0 und die Funktionen **RLOCK(), FLOCK()** und **LOCK()** liefern immer .T., das heißt, der ausschließliche Zugriff wurde gewährt. Dadurch ist es selbst mit diesen nicht netzfähigen Installationen von dBase III Plus und ohne die entsprechende Hardware möglich, Programme zu schreiben, die für den Mehrnutzerbetrieb ausgelegt sind.

Insgesamt stellt dBase III Plus einen bemerkenswerten Fortschritt gegenüber dBase III dar, der einen Umstieg lohnt. Das gilt ungeachtet aller kritischen Bemerkungen, denn es wäre verwunderlich, wenn mit der Verfügbarkeit eines Programms und seiner praktischen Nutzung nicht sofort weitere Wünsche und Ideen einer besseren Realisierung entstehen würden.

Literatur

/1/ Grafik, W.; Osten, B.: dBase III im Vergleich. Mikroprozessortechnik, Berlin 2 (1988) 10, S. 294

Computer-Club

64-KByte-RAM-Module als Datenspeicher für Basic-Programme im KC 85/3

Eckehard Jagdmann, Dr. Werner Domschke VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen

Ein wesentlicher Vorteil des KC 85-Systems des VEB Mikroelektronik „Wilhelm Pieck" ist das modulare Konzept. Insbesondere die Möglichkeit der Erweiterung des Speichers über den 64-KByte-Adreßraum des Prozessors U 880 D ist hervorzuheben. Als Grundlage dafür befindet sich der Modul M011 64 KByte RAM in Produktion. Im KC 85/2,3 können davon ohne weitere Aufsätze 2 Stück betrieben werden. Damit stehen dem Anwender insgesamt 144 KByte RAM im Grundgerät zur Verfügung. Dieser umfangreiche Speicher kann in Maschinensprache oder in Forth relativ problemlos genutzt werden. In Basic bereitet das größere Probleme, da der Programm- und Datenspeicher durch den Interpreter dynamisch verwaltet werden. Der Anwender kann ohne spezielle Maßnahmen nicht eindeutig festlegen, auf welcher Speicheradresse bestimmte Daten stehen. Damit sind von dem 64-KByte-RAM-Modul nur 32 KByte nutzbar.

Im folgenden soll eine Möglichkeit aufgezeigt werden, wie auch in Basic mit dem KC 85/3-Grundgerät 143 KByte (15 KByte für Programm und 128 KByte für Daten) bzw. bei Ausbau des Systems mit einem oder mehreren Aufsätzen auch größere Speicher genutzt werden können.

Datenspeicherung in Basic

Der Basic-Interpreter richtet in dem ihm zur Verfügung stehenden Speicher nach **RUN** und der Benutzung von Variablen bzw. deren expliziter Vereinbarung (**DIM**-Anweisung) Variablenbereiche ein. Dabei werden zwei Bereiche unterschieden:

1. Bereich der einfachen numerischen und Stringvariablen sowie der definierten Funktionen (**DEFFN**-Anweisung)
2. Bereich der numerischen und Stringfelder.

Aus den Arbeitszellen des Basic-Interpreters läßt sich die Lage der Bereiche eindeutig feststellen:

Adresse	Inhalt
3D7H	Anfang des 1. Bereiches
3D9H	Anfang des 2. Bereiches
3DBH	erster freier Speicherplatz nach dem 2. Bereich

Die Variablen werden wie folgt gespeichert:

Einfache numerische Variablen

2 Byte – erste 2 Buchstaben des Namens (vertauscht)
4 Byte – Wert der Variablen

Einfache Stringvariablen

2 Byte – erste 2 Buchstaben des Namens (vertauscht; Bit 7 des ersten Bytes gleich 1)
2 Byte – Länge des Strings
2 Byte – Anfangsadresse des Strings
Die Zeichenkette selbst kann im Stringspeicher oder im Programm enthalten sein.

Numerische Felder

2 Byte – erste 2 Buchstaben des Namens (vertauscht)
2 Byte – Anzahl der nachfolgend zum Feld gehörenden Bytes
1 Byte – Anzahl der Dimensionen des Feldes
(1 – Vektor; 2 – Matrix; 3 – dimensionales Feld usw.)
je 2 Byte – Anzahl der Feldelemente je Dimension
je 4 Byte – Wert des Feldelements

Stringfelder

2 Byte – erste zwei Buchstaben des Namens (vertauscht; Bit 7 des ersten Bytes gleich 1)
2 Byte – Anzahl der nachfolgend zum Feld gehörenden Bytes
1 Byte – Anzahl der Dimensionen des Feldes
je 2 Byte – Anzahl der Feldelemente je Dimension
je 4 Byte – Beschreibung des Strings (2 Byte Länge; 2 Byte Anfangsadresse des Strings)

Plazierung von Daten außerhalb des Basic-Speichers

① Zugriff auf den 64-KByte-Modul von Basic aus

Das Problem bei der Einbindung von 64-KByte-RAM-Modulen in Basic-Programmen besteht darin, daß Daten in einen Speicher abgelegt werden müssen, der nicht vom Basic-Interpreter verwaltet wird und dessen Anordnung im Adreßraum des Prozessors während der Programmabarbeitung veränderlich ist.

Die Speichergliederung im KC 85/3-System mit einem 64-KByte-RAM-Modul mit den 4 möglichen Speicherblockanordnungen ist im Bild 1 dargestellt. Daraus wird deutlich, daß der Basic-Arbeitsspeicher auf den 16-KByte-RAM-Block des Grundgerätes beschränkt werden muß, wenn die Blöcke des 64-KByte-Moduls während der Programmabarbeitung vertauscht werden sollen. Weiterhin erkennt man, daß in der Speicherlücke zwischen RAM und IRM auf den 64-KByte-Modul zugegriffen werden kann (Adresse 4000H bis 7FFFH).

Da der IRM vom Basic-Interpreter ständig in den Hintergrund geschaltet wird, ist auch der Bereich von 8000H bis 0BFFFH dafür nutzbar.

② Datenspeicherung im 64-KByte-Modul

Bei einfachen numerischen Daten bieten sich die Basic-Anweisungen **PEEK** und **POKE** für Bytes (Zahlenbereich 0...255) bzw. **DEEK** und **DOKE**

	0	16K	32K	48K
SWITCH 8,3	0	1	2	3
SWITCH 8,67	1	0	3	2
SWITCH 8,131	2	3	0	1
SWITCH 8,195	3	2	1	0
	RAM		IRM	Basic (48K–56K) / CAOS (56K–64K)

Bild 1 Anordnung der Speicherblöcke des 64-KByte-Moduls im Adreßraum des KC 85/3 bei unterschiedlicher Modulprogrammierung

```
10 GOTO1000
100 !### UP1-BLOCK/SWITCH ###
110 A=INT(BL/BZ)
115 BL=BL-A*BZ : A=A*4+SA : FOR I=0 TO MA-1 : SWITCH(I*4+SA),0 : NEXT
120 SWITCHA,INT(BL/BA)*64+3
130 BL=BL-INT(BL/BA)*BA
140 RETURN
200 !### UP2-ZAHL/SWITCH ###
210 BL=INT(ZZ*2/SL) : ZA=ZZ-BL*SL/2 : GOSUB100
220 ZA=SG+BL*SL+ZA*2
230 RETURN
300 !### UP3-STRING->RAM ###
310 GOSUB100 : RA=SG+SL-1+SL*BL
320 L=LEN(B$)
330 IF L<SL THEN B$=B$+STRING$(SL-L," ")
340 IF L>SL THEN B$=LEFT$(B$,SL)
350 HV=DEEK(SP)
360 DOKESP,RA
370 DA$=B$
380 DOKESP,HV
390 RETURN
400 !### UP4-RAM->STRING ###
410 GOSUB100
420 DOKEFNVA(2),FNAD(BL) : !ANFANGSADRESSE IM RAM
430 POKEFNVA(1),SL : !LAENGE DES STRINGS
440 B$=DA$ : RETURN
500 !### UP5-ZAHL->RAM ###
510 GOSUB200 : !0<=WERT<=65535
520 IF WERT>32767 THEN WERT=WERT-65536
530 DOKEZA,WERT
540 RETURN
600 !### UP6-RAM->ZAHL ###
610 GOSUB200 : !0<=WERT<=65535
620 WERT=DEEK(ZA) : IF WERT<0 THEN WERT=WERT+65536
630 RETURN
700 !### UP7-INITIALISIERUNG ###
710 DA$=" " : VA=983 : SP=964
720 DEFFNDA(X)=DEEK(DEEK(VA)+4+X)
730 DEFFNVA(X)=DEEK(VA)+2*X
740 DEFFNDL(X)=PEEK(DEEK(VA)+2+X)
750 DEFFNAD(X)=X*SL+SG
760 FOR A=523 TO 740 : READB : POKEA,B : NEXT
770 SG=16384 : !GROESZE EINES SPEICHERBLOCKS
780 SL=64 : !STRINGLAENGE
790 BA=SG/SL : !ANZAHL DER STRINGS PRO SPEICHERBLOCK
800 MA=2 : !MODULANZAHL
810 BZ=4*BA : !ANZAHL DER STRINGS PRO MODUL
820 SA=8 : RETURN : !STECKPLATZ DES 1.MODULS
1000 ! #### MENUE ####
1010 GOSUB700
1020 CLS : PRINT"STRINGVERWALTUNG...S"
1030 PRINT"ZAHLENVERWALTUNG...Z"
1040 PRINT"RETTEN & LADEN.....R" : PRINT
1050 A$=INKEY$ : IFA$=""THEN1050
1060 IF A$="S" THEN2000
1070 IF A$="Z" THEN3000
1080 IF A$="R" THEN4000 ELSE1050
2000 !#### STRINGVERWALTUNG ####
2010 PRINT"SCHREIBEN BLOCK...S"
2020 PRINT"LESEN BLOCK.......L"
2030 PRINT"MENU..............M"
2040 A$=INKEY$ : IFA$="" THEN2040
2050 IF A$="S" THEN2100
2060 IF A$="L" THEN2200
2070 IF A$="M" THEN1020 ELSE2040
2100 INPUT"INDEX";BL : IF BL<0 OR BL>(BZ*MA-1) THEN2100
2110 INPUT"TEXT";B$
2120 GOSUB300 : GOTO2000
2200 INPUT"INDEX";BL : IF BL<0 OR BL>(BZ*MA-1) THEN2200
2210 GOSUB400 : PRINTDA$ : GOTO2000
3000 !#### ZAHLENVERWALTUNG ####
3010 PRINT"SCHREIBEN ZAHL...S"
3020 PRINT"LESEN ZAHL.......L"
3030 PRINT"MENU.............M"
3040 A$=INKEY$ : IF A$="" THEN3040
3050 IF A$="S" THEN3100
3060 IF A$="L" THEN3200
3070 IF A$="M" THEN1020 ELSE3040
3100 INPUT"INDEX";ZZ : IF ZZ<0 OR ZZ>(SG*2*MA-1) THEN3100
3110 INPUT"WERT";WERT
3120 GOSUB500 : GOTO3000
3200 INPUT"INDEX";ZZ : IF ZZ<0 OR ZZ>(SG*2*MA-1) THEN3200
3210 GOSUB600 : PRINT"WERT=";WERT : GOTO3000
4000 !#### LADEN & RETTEN ####
4010 PRINT"RETTEN...R" : PRINT"LADEN....L" : PRINT"MENU.....M"
4020 A$=INKEY$ : IF A$="" THEN4020
4030 IF A$="R" THEN4100
4040 IF A$="L" THEN4200
4050 IF A$="M" THEN1020ELSE4020
4100 INPUT"STECKPLATZ";A : POKE514,A
4110 A=1 : INPUT"AB WELCHEN BLOCK (1/2/3/4)";A : POKE526,A
4120 INPUT"BYTEANZAHL";A : IF A>65536 THEN4120
4130 IF A>32767 THEN A=A-65536
4140 DOKE512,A : INPUT"NAME";A$
4150 IF LEN(A$)<7 THEN A$=A$+STRING$(7-LEN(A$)," ")
4160 FOR A=1 TO 7 : POKE514+A,ASC(MID$(A$,A,1)) : NEXT
4170 FOR I=0 TO MA-1 : SWITCH(I*4+SA),0 :NEXT : CALL*247 : GOTO4000
4200 INPUT"STECKPLATZ";A : POKE514,A
4210 A=1 : INPUT"AB WELCHEN BLOCK (1/2/3/4)";A : POKE526,A
4220 A=4 : INPUT"WIEVIEL BLOECKE MAXIMAL (1/2/3/4)";A : POKE527,A
4230 FOR I=0 TO MA-1 : SWITCH(I*4+SA),0 : NEXT
4240 CALL*211 : GOTO4000
5000 DATA 68,65,84,1,4
5010 DATA 0,205,24,240,205,189,2,58,15,2,60,50,15,2,62,0
5020 DATA 50,16,2,50,129,183,58,15,2,61,50,15,2,40,11,205
5030 DATA 3,240,16,58,16,2,254,1,40,228,205,27,240,201,62,1
5040 DATA 50,16,2,205,212,2,201,205,24,240,205,189,2,42,0,2
5050 DATA 43,62,49,50,10,2,1,0,64,175,60,237,66,210,90,2
5060 DATA 183,237,74,32,1,35,9,34,0,2,50,16,2,254,2,56
5070 DATA 49,62,3,50,129,183,33,0,64,34,130,183,33,255,127,34
5080 DATA 132,183,33,62,2,34,134,183,33,3,2,205,3,240,54,58
5090 DATA 10,2,60,50,10,2,205,212,2,58,16,2,61,50,16,2
5100 DATA 24,203,62,2,50,129,183,33,0,64,34,130,183,42,0,2
5110 DATA 34,132,183,33,3,2,205,3,240,54,195,58,2,58,2,2
5120 DATA 111,58,14,2,71,62,195,22,64,130,16,253,87,62,2,205
5130 DATA 3,240,38,201,58,2,2,33,0,184,133,111,126,38,64,132
5140 DATA 87,62,2,24,234
```

für Worte (2 Byte, Zahlenbereich −32768...32767) an. Die Organisation im 64-KByte-Speicherblock erfolgt in der Art, daß der zur Verfügung stehende Speicher (Anzahl der 64-KByte-Module) als eindimensionales Feld aufgefaßt wird. Der Zugriff auf ein Feldelement (lesen oder schreiben) muß so organisiert werden, daß
1. der entsprechende 64-KByte-Modul eingeschaltet und der erforderliche Speicherblock in dem Adreßbereich 4000H...7FFFH mit der **SWITCH**-Anweisung angeordnet wird
2. die aktuelle Speicheradresse im Block berechnet wird. Danach kann mit **DEEK** oder **DOKE** die Zahl gelesen oder geschrieben werden. Im Listing ist ein Beispielprogramm angegeben, das dieses in den Unterprogrammen Zeile 100 bis 630 demonstriert.
Sollen Stringvariablen in Speicherblöcken des 64-KByte-Moduls abgelegt werden, so kann die Arbeit durch Beeinflussung von Basic-Arbeitszellen effektiv gestaltet werden. In den Arbeitszellen des Interpreters existiert ein Pointer, in dem die nächste freie Adresse im Stringraum des RAM steht (3C4H). Dort wird bei jedem Belegen einer Stringvariablen die Zeichenkette abwärts im Speicherraum eingetragen. Die nächste freie Adresse steht danach wieder in 3C4H. Soll eine Zeichenkette in einen 64-KByte-Modul übergeben werden, braucht nur die gewünschte Adresse innerhalb des mit **SWITCH** zugewiesenen 16-KByte-RAM-Blocks dem Stringpointer (3C4H) übergeben werden. Danach wird bei einer Stringmanipulation (Zeile 370) der neue String ab dieser Adresse abwärts eingetragen. Ist die Zeichenkette übergeben, muß wieder die Adresse in 3C4H eingetragen werden, die vor der Manipulation darin stand.
Sollen Strings verarbeitet werden, die in einem Block des 64-KByte-Moduls stehen, so ist das ebenfalls relativ einfach möglich. Es wurde bereits darauf hingewiesen, daß im Variablenspeicher die Stringvariablen mit Name, Länge und Anfangsadresse des Strings im RAM abgelegt sind. Zur Übernahme einer Zeichenkette aus dem 64-KByte-Modul ins Basic-Programm braucht nur die Anfangsadresse des Strings im zugewiesenen RAM-Block des 64-KByte-Moduls und dessen Länge der gewünschten Stringvariablen im Variablenspeicher übergeben zu werden.

Programmbeschreibung
In Bild 2 ist ein Beispielprogramm angegeben, mit dem die vorgestellten Funktionen erfüllt werden. Der universelle Teil des Beispiels ist in Form von Unterprogrammen ausgeführt (Zeilennummer 100 bis 820). In den Zeilen 1000 bis 3210 ist als Beispiel die Arbeit mit den Unterprogrammen dargestellt, und in dem letzten Teil (Zeilennummer 4000 bis 5140) eine Möglichkeit zum Retten und Laden der Daten angegeben. Diese Teile werden nachfolgend kurz beschrieben.

① Basic-Unterprogramme
Der eigentliche Kern des Beispielprogramms ist der Block mit den Unterprogrammen (Zeilen 100 bis 630). In ihnen findet der Austausch der Daten zwischen Basic-Programm und RAM der Module M011 statt. Die Unterprogramme sind auf die Verwaltung von Zeichenketten der Länge 64 Byte bzw. Zahlen von 2 Byte ausgelegt. Werden Elemente anderer Länge verwendet, sind die im Programm in den Zeilen 770 bis 810 angegebenen Konstanten entsprechend zu ändern.
– Zeile 100–140 (UP1)
Hier wird aus dem Index BL der Elemente (64 Bytes lange Zeichenketten) der zuständige M011 und der 16-KByte-Block im M011 errechnet und in den Adreßbereich von 4000H-7FFFH zugewiesen. Für die weitere Adreßberechnung wird der Index BL aufbereitet.
– Zeile 200–230 (UP2)
Um UP1 auch für 2 Byte lange Elemente nutzen zu können, wird vor dem Aufruf von UP1 aus dem Index ZZ der Index BL gebildet. Mit ZA steht nach Abarbeitung von UP2 die Anfangsadresse des Elementes im Speicher zur Verfügung.
– Zeile 300–390 (UP3)
In UP3 wird die Länge des in den RAM zu übergebenden Strings B$ auf 64 Zeichen festgelegt. Danach erfolgt die Berechnung der Adresse im 16-KByte-RAM-Block, ab der der String B$ abwärts eingetragen werden soll. Durch Übergabe dieser Adresse in den Stringpointer (3C4H) und nachfolgender Stringmanipulation (Zeile 370) wird die Zeichenkette dem 64-KByte-Modul übergeben. Der Stringpointer wird vor dem Unterprogrammrücksprung wieder zurückgestellt.
– Zeile 400–440 (UP4)
Soll ein Stringelement aus einem M011 ins Basic-Programm übernommen werden, wird nur die Anfangsadresse der Zeichenkette im Speicher und deren Länge dem Variablenspeicher übergeben. In UP 4 wird beides realisiert. Nach Aufruf von UP4 steht in DA$ die ausgelesene Zeichenkette zur Verfügung.
– Zeile 500–530 (UP5)
In UP5 werden 2-Byte-Integerzahlen in einen M011 geschrieben. Dazu wird dem Unterprogramm der Index ZZ und der Wert der Zahl (in *WERT*) übergeben.
– Zeile 600–630 (UP6)
Mit diesem Unterprogramm wird eine 2-Byte-Zahl aus dem RAM gelesen. Dazu wird der Index ZZ an UP6 übergeben. Nach der Abarbeitung steht in *WERT* der ausgelesene Zahlenwert.
– Zeile 700–820 (UP7)
In UP7 werden Funktionen definiert, die in den anderen Unterprogrammen zur Adreßrechnung benötigt werden. Deshalb muß dieses Unterprogramm beim Start des Basic-Programms zuerst aufgerufen werden. Weiterhin wird das in einer **DATA**-Liste vorhandene Maschinenprogramm, zum Laden und Retten der Dateien, in den Speicherbereich von 20BH bis 2E4H geschrieben, und es werden die speziellen Konstanten zur Modifikation des Programms festgelegt.
② Demonstrationsteil
● Zahlenverwaltung (Zeile 3000–3210)
In diesem Programmteil kann bei der Verwendung von zwei Modulen M011 ein Feld von 65536 Zahlen (2 Byte lang) verwaltet werden. Es können Zahlen über Tastatur eingegeben oder auf dem Bildschirm angezeigt werden. Dazu ist jeweils der Index einzugeben.
● Zeichenkettenverwaltung (Zeile 2000–2210)
Unter Verwendung von zwei Modulen M011 können in diesem Teilprogramm 2048 Strings der Länge 64 Byte eingegeben und/oder angezeigt werden. Bei der Eingabe der Zeichenkette braucht die Länge nicht exakt 64 Zeichen zu betragen. In den Unterprogrammen werden die eingegebenen Strings auf 64 Zeichen Länge korrigiert.

③ Laden und Retten von Dateien
Von Basic aus besteht nicht die Möglichkeit, bestimmte Speicherbereiche auf Kassette auszulagern. Um die Daten von den Modulen M011 64 KByte RAM retten zu können, ist eine kurze Maschinenroutine notwendig.
Das Maschinenprogramm belegt den Speicher von 20BH bis 2E4H. In ihm ist gleichzeitig eine Laderoutine enthalten. Beide Teilprogramme können von Basic mit **CALL∗** aufgerufen werden. Es kann maximal der gesamte Speicherbereich eines M011 mit einem Aufruf auf Kassette gerettet werden. Werden mehrere M011 verwendet, muß dementsprechend mehrmals gerettet werden. Dazu sind dem Unterprogramm jeweils folgende Daten zu übergeben:
● Steckplatz des M011 in der Speicherzelle 202H; **POKE**514,x
● 16-KByte-Block (1–4), ab dem die Daten gerettet werden sollen; **POKE**526,y
● zu rettende Byteanzahl im M011 (maximal 65536) in den Speicherzellen 200H und 201H; **DOKE**512,z
● 7stelliger Name von 203H bis 209H
Sollen Daten aus einem M011 im Modulschacht 12 gerettet werden, ist der im Schacht 8 vorhandene Modul M011 inaktiv zu schalten. Der Aufruf *Retten* erfolgt mit **CALL∗**247.
Beim Laden müssen dem Unterprogramm folgende Daten übergeben werden:
● Steckplatz des M011 in der Speicherzelle 202H; **POKE514,x**
● Anzahl der maximal zu ladenden 16-KByte-RAM-Blöcke (1–4) in 20FH; **POKE**527,y
● 16-KByte-RAM-Blocknummer (1–4), ab der die Daten geladen werden sollen; **POKE**526,z
Der Aufruf *Laden* erfolgt mit **CALL∗**211. Wenn Daten in den 64-KByte-RAM-Modul im Schacht 12 geladen werden sollen, muß der Modul im Schacht 8 inaktiv geschaltet werden.
Werden mehr als 2 Module M011 verwendet (mit D002 BUSDRIVER), müssen beim Laden bzw. Retten in/aus Modulen mit höherer Steckplatznummer alle Module M011 in den Schächten mit niedrigerer Nummer inaktiv geschaltet werden.
Im Programmabschnitt (Zeilen 4000–4240) werden die Parameter dem Maschinenunterprogramm übergeben. Danach wird dann mit **CALL∗** der gewünschte Programmteil aufgerufen.
Die Anzahl der maximal zu ladenden Blöcke ist standardmäßig auf 4 festgelegt. Sollte aber die gerettete Datei zum Beispiel 3 Blöcke lang sein, und es sollen aber nur 2 Blöcke davon wieder geladen werden, ist der Speicherzelle 20FH eine 2 zu übergeben. Weiterhin muß beim Retten oder Laden die Nummer des Blockes angegeben werden, ab dem gerettet oder geladen werden soll. Fehlt diese Angabe, ist bei Neustart des Programms immer standardmäßig der Block 1 eingestellt.
Der Dateityp der auf die Kassette geretteten Dateien ist DAT. Diese Dateien können nur wieder mit diesem Maschinenunterprogramm eingelesen werden.

✉ KONTAKT ☎

VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Eisenacher Straße 40, Mühlhausen, 5700; Tel. 5 32 88 (E. Jagdmann)

Abwechslung im Menü

Moderne Benutzeroberflächen bieten unter Einbeziehung von Joysticks oder Maus komfortablen Computerdialog. Auch bei kleineren Rechnern, mit geringeren Hardwaremöglichkeiten, sollte über Menü- oder Fenstertechnik solcherart Softwareergonomie angestrebt werden. Da ein Joystickanschluß bisher auf den KC 87 begrenzt ist, soll im folgenden eine aufgelockerte Menügestaltung für den KC 85/2 und /3 in Form eines Laufschriftbalkens aufgezeigt werden.
Oft ermöglichen Auswahlmenüs Alternativsprünge in Unterprogramme über Tastatureingabe der jeweiligen Kennzahl. In BASIC erfordert dabei der INPUT-Befehl nach beendeter Eingabe die Entertaste, was im folgenden Gerüst für einen BASIC-Trainer durch INKEY$ entfällt. Die Lösung ist optisch ansprechend und hinreichend modifizierbar.

```
10   CLS:REM LAUF-DEMO
20   ?AT(3,5);"1...FELDER"
30   ?AT(4,5);"2...STRINGS"
40   ?AT(5,5);"3...FARBEN"
 .
 .

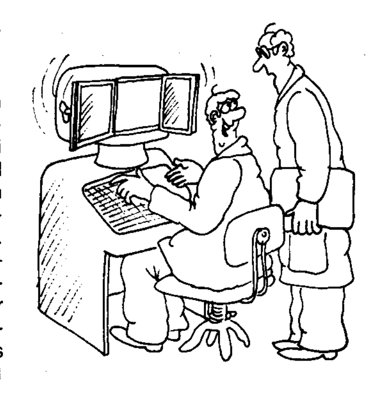
61   W$="+BITTE ENTSCHEIDEN
     SIE SICH"
62   N$=W$+W$
63   Z=Z+1
64   ?AT(25,8);MID$(N$,Z,20)
65   IF Z=LEN(W$) THEN Z=0
66   E$=INKEY$:IF E$="" GOTO 63
67   REM VERZWEIGUNG: CLS
68   ON VAL(E$) GOSUB 100,500,900
69   RUN
100  REM ARBEIT MIT FELDERN
 .
 .

500  REM STRINGOPERATIONEN
 .
 .

900  REM FARBGRAFIK AM KC 85/3
```

Gerhard Obenaus

Ein Fenster, was sonst?



Zeichnung Steger

Konvertierung MEOS-SCP

Mit zunehmender Ausrüstung der Softwareentwickler mit PC/BC standen die Probleme, MRES-Software im MEOS-Format dort weiterzunutzen bzw. auf den PCs/BCs Programmentwicklung für MEOS-Anwender zu realisieren.
Zu diesem Zweck wurden Programmmodule für PC/BC unter SCP entwickelt, die auf den Ebenen Quell-, Text- und auch Maschinenprogramme eine Diskettendateikonvertierung zwischen SCP und MEOS in beiden Richtungen ermöglichen. Hardwarevoraussetzung für den Konvertierungsrechner ist der Anschluß mindestens eines Standard-Floppy-Laufwerkes (8") für die MEOS-Schnittstelle.
Als erstes Ergebnis dieser Entwicklung wurde z. B. für den BC 5120 eine Version des Echtzeitbetriebssystems EIEX mit Zusatzmodulen zur Diskettenarbeit im SCPX-Format entwickelt. Die Konvertierungsprogramme liegen sowohl als Maschinenprogramme als auch als Quellprogramme mit einer Minimaldokumentation zur Nachnutzung vor.

SDAG Wismut, WTZ, Abt. Absatz, Karl-Marx-Straße 13, Grüna, 9125

Hauschild

Grafikleiterkarte im K 1520-Format

Im VEB Weimar-Werk wurde eine Grafikleiterkarte mit dem GDC U 82720 (siehe MP 4/1987, Seite 99) entwickelt. Sie ist speziell für das Programmiergerät PRG 700 vom VEB Numerik Karl-Marx-Stadt im K 1520-Kartenformat ausgelegt, kann aber auch in allen anderen Rechnern, die auf K 1520-Basis arbeiten, eingesetzt werden.
Grundlage für den Einsatz ist das Betriebssystem SCP. Mit dieser Karte können in den Grenzen der 8-Bit-Technik
- Technische Zeichnungen erstellt
- alphanumerische Zeichen in verschiedenen Varianten geschrieben
- der Zoomfaktor gewählt
- der Displaybereich verschoben
- Makros definiert werden.

Der Ausdruck ist durch Plotter bzw. grafikfähige Drucker über eine V.24-Schnittstelle vorgesehen.
Für diese Karte liegen nachnutzungsfähige Unterlagen vor. Interessenten für eine DKL bitten wir, kurzfristig eine Bestellung auszulösen, um eine Fertigung realisieren zu können.

VEB Weimar-Werk, Büro für Neuererwesen/Abteilung REA, PSF 305, Weimar, 5300; Tel. 71 32 11

Fischer

Datenbankbetriebssystem RELA 1.4/T

RELA wird seit dem 1.6.1988 auf KC 85/3 in verschiedenen Bereichen des privaten Handwerks eingesetzt.
Die RAM-Floppy des Datenbanksystems umfaßt 32 KByte für die Aufnahme von Anwenderprogrammen. Zum Lieferumfang gehört eine detaillierte Dokumentation, Konsultation und Systemanpassung, ein Auto-Ladesystem, die Eröffnung einer Programmbibliothek und der aktuelle Softwarekatalog.
Die wichtigsten Merkmale:
- max. 528 KByte Dateispeicher im Grundgerät (272 KByte bei Druck)
- beliebige Erweiterungen bei Nutzung eines Aufsatzes und von M011-Erweiterungsmodulen
- Dateien zu je 32 KByte, TEXOR-kompatibel
- Auto-Ladesystem, Turbo mit 9124 Baud
- Festkommaarithmetik, max. 20stellig
- kundenspezifische Druckerschnittstelle, verschiedene Druckmodi als Systemfunktion (Listen, Tabellen)
- Sortier-, Such-, Editier- u.a. Systemfunktionen durch Menü- und Fenstersteuerung zu erreichen.

Folgende Problemlösungen existieren:
- Rechnungslegung: wahlweise mit Materialbestandskorrektur, Ausschrift und Berechnung der Regelpreise, Auftragseingabe oder Stammdatei sowie Ablage und Übertragung in ein Journal
- Inventur 1: nur Summenblatt oder komplette Auflistung
- Inventur 2: Zähllisten und Inventur mit Soll-Ist-Vergleich
- Kassenbuch: Tageskasse, Monatsabrechnung, Kumulation
- Lagerhaltung: Bestellung, Zugänge Bestandsermittlung, periodische und kumulative Listen
- Kontoprüfung: Belegabarbeitung im Buchhandel
- Sonderprogramme.

ELECTRONIC-SERVICE Helmut Ulbricht, Göttiner Landstraße 7, Brandenburg, 1800; Tel. 52 20 76

Zierott

Softwareansteuerung von Druckern unter DCP 1700

Das Dienstprogramm **Multiprint** wurde zur komfortablen Steuerung der Epson-Drucker FX 800/FX 1000 im Geräteverbund mit dem AC 7150 unter DCP 1700 entwickelt. Es liegt zur Zeit auch in einer Version für die Epson-Drucker LQ 850/LQ 1050 (unter MS-DOS 3.2.) vor. Das Programm weist folgende Merkmale auf:
- Realisierung von über 40 Steuerbefehlen durch übersichtliche Menüsteuerung (in Fenstertechnik, mit Cursortasten bzw. Maus)
- Bereits beim Aufruf realisiert das Programm unabhängig von der DIP-Schalterstellung des Druckers eine erfahrungsgemäß günstige Grundinitialisierung mit Voreinstellung der Menüs auf die am häufigsten genutzten Optionen.
- Das Programm ist selbsterklärend und kann auch durch Ungeübte sofort bedient werden.
- Als besonderen Komfort weist das Programm die Möglichkeit auf, Textfiles direkt ausdrucken zu können, was die Verwendung eines Textverarbeitungssystems in vielen Fällen unnötig macht.
- Fehler bei der Eingabe bzw. durch nicht bereite Hardware werden durch eine programmeigene Fehlerroutine abgefangen.

In der Anwendung als günstig erweist sich die Einbindung des Programms in eine geeignete Nutzeroberfläche wie das Usermenü des Norton-Commanders, so daß es aus allen Arbeitsebenen heraus angesprochen werden kann.
Das Programm ist in Turbo-Pascal realisiert und liegt einschließlich Dokumentation als ausführbares COM-File zur Nachnutzung vor. Eine Anpassung an andere Druckertypen, die ebenfalls Epson-Steuercodes verwenden, ist möglich.

Komische Oper Berlin, EDV/BO, Behrenstraße 55–57, Berlin, 1080; Tel. 2 20 27 61/2 24

Balzer

Kopplung von V.24 und seriellem IEC-Bus über Multiplexer

Der *Multiplexer* ermöglicht die Kopplung, auch im Sinne einer Peripherieerweiterung, von beliebigen Rechnern mit einem V.24-Interface (RS-232-C) und Geräten mit dem seriellen IEC-Bus-Interface, das beispielsweise Commodore-Geräte (C128, C64, C16), deren Peripherie (Floppy 1541, Drucker MPS 803...) oder ungarische Heimcomputer (Primo) besitzen. Neben dem *Multiplexer* existiert eine ausführliche Beschreibung (Dokumentation) des seriellen IEC-Busses bestehend aus:
- verbaler Beschreibung
- Zeitdiagrammen
- Zustandsgraphen
- Programmablaufplänen und Assembler-Quellcode (Z 80).

Damit lassen sich auch wesentlich einfachere Lösungen, z.B. Anpassung von COMMODORE-Peripherie an *Z 80-Rechner* oder Übertragung von Daten zwischen C16, C64... und anderen Rechnern, finden.

VEB EAW Elektronik Dresden, Abt. WEL, Großenhainer Straße 1–7, Dresden, 8060; Tel. 5 94 82 34

Seifert

Programmverwaltung auf KC 85/3

Das Programm **DATVER** dient der Verwaltung von Basic-Programmdateien. Hierzu wird am Anfang einer jeden Programmkassette eine Datei DIREKTORY abgelegt. Nach Programmstart und Einladen einer Direktory werden die Programmbezeichnungen angezeigt. Der Hinweis auf die MB-Startposition erfolgt nach Eingabe des gewünschten Programmnamens. Nach Positionierung des Bandes und Tastendruck erfolgt das Einladen des Programms, nach erneutem Tastendruck die Abarbeitung desselben. Bei Beendigung der Abarbeitung verzweigt DATVER zur Anzeige der Programmbezeichnungen, und das nächste Programm bzw. eine andere Direktory können geladen werden. Das dazugehörige Dienstprogramm EKDIR ermöglicht den Aufbau sowie die Pflege der Datei DIREKTORY. Es werden die Funktionen Einlesen, Abspeichern, Anzeigen, Hinzufügen, Korrektur und Löschen realisiert.
Für die Anwenderprogramme gelten folgende Einschränkungen: Kleinste Zeilennummer = 19500 (das kann mit einer RENUMBER-Anweisung leicht realisiert werden) und am Programmende muß Sprung zur Zeile 9000 erfolgen (vor Programmende GOTO 9000 einfügen).

VEB Leichtmetallgußwerk, Abt. EDV, Enge Gasse, Bad Langensalza, 5820

Dünnebeil

Hexadezimaleditor

Zur Nachnutzung bieten wir für den Mikrorechnertyp AC 7100 unter dem Betriebssystem SCP 1700 den Hexadezimaleditor **HEXER** an. Mit HEXER lassen sich einfache Editorfunktionen wie Löschen, Einfügen und Überschreiben auf beliebige Daten anwenden. Pro Bildschirm werden 128 Bytes im hexadezimal- und ASCII-Format dargestellt, wobei der Editiermodus zwischen beiden Formen beliebig umschaltbar ist. Die Kommandosyntax wurde an Wordstar angelehnt; die Positionierung erfolgt mittels Kursortasten oder über ein integriertes Suchkommando. Mit HEXER können sowohl Dateien als auch direkt adressierte Einzelblöcke bearbeitet werden; Einzelblöcke lassen sich auch zu einer Datei zusammenfassen.
HEXER besitzt nicht die Leistungsfähigkeit professioneller Editoren, ist aber besonders bei Reparaturarbeiten, z.B. an REDABAS-Dateien, sehr nützlich.

VEB LMB Güstrow, Abt. L 102, Röverlannen, Güstrow, 2600; Tel. 4 65 15

Ansorge

GEM-Interface für Turbo-Pascal 4.0

Es wurde eine Anbindung von Turbo-Pascal (Version 4) an GEM entwickelt. Sie erlaubt den Einsatz der AES- und VDI-Bibliotheksfunktionen in eigenen, in Turbo-Pascal geschriebenen Programmen.
AES (Application Environment Services) ist ein System zur Steuerung von grafischen Eingaben. Es enthält die Routinen für verschiebbare Fenster, Pull-down-Menüs, Warn- und Alarmboxen, Systemrufe und Möglichkeiten zur Reaktion auf Maus- und Tastatureingaben.
VDI (Virtual Device Interface) ist eine geräteunabhängige Grafikschnittstelle. VDI ist ein stark erweiterter Nachfolger von GSX. Grafikprogramme und grafische Bestandteile beliebiger Programme können mit VDI geräteunabhängig erstellt werden.
Die nachnutzbare Softwareleistung besteht aus folgenden Komponenten:
Unit **AESGEM** – AES-Funktionen
Unit **VDIGEM** – VDI-Funktionen
Unit **GEMDEF** – Konstanten, Variablen und Hilfsfunktionen, die den Einsatz der beiden anderen Units erleichtern.
Alle Komponenten werden in übersetzter Form und als Quelltext übergeben. Einsetzbar ist die Einbindung auf dem AC 7150 und IBM-kompatiblen PCs mit Grafikkarte. Die Einbindung ist lauffähig ab GEM-Version 1.x. Es wurden alle Funktionen realisiert, auch die ab Version 2.x nicht mehr unterstützten.
Zusätzlich wird eine ausführliche Be-

schreibung (auf Diskette) der AES-Funktionen zur Nachnutzung angeboten. Eine Beschreibung der VDI-Funktionen ist in Vorbereitung.

Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Direktorat R, Wissenschaftsbereich Informationsverarbeitung, Richard-Wagner-Straße 31, Warnemünde, 2530

Lübcke

Textverarbeitungssystem SALOTEX '87 für ATARI 800 XL und KC 85/3

SALOTEX wurde als Betriebssystemerweiterung des ATARI 800 XL mit der Zielstellung konzipiert, ein kassettenorientiertes Textverarbeitungssystem zu schaffen, das vollständig kompatibel mit dem modularen System des KC 85/2, /3 ist und sich damit optimal in das vorhandene Hardwareangebot der DDR-Produktion einpaßt.

Die nachfolgend skizzierten Charakteristika von SALOTEX deuten an, inwiefern diese Zielstellung erreicht wurde:

– Verfügbarer Textspeicher von 39 bis 44 KByte (etwa 25 bis 30 Schreibmaschinenseiten zu 30 Zeilen mit 60 Anschlägen)
– Wahl zwischen 40- und 64-Zeichendarstellung auf 24 bzw. 32 Zeilen (bei druckbildadäquater Raumaufteilung)
– Kontinuierliches Zeilenscrolling im gesamten Text (verzögerungsfrei vor- und rückwärts) und seitenweises Blättern
– Kompatibilität der Schreib-/Leseroutine zum KC 87 bzw. KC 85/3 sowie der Texte zum TEXOR-Modul des KC 85/3
– Zusätzliche Turbo-Schreib-/Leseroutine mit der 10fachen Bitrate gegenüber der Standardroutine des ATARI 800 XL
– Reversible Formatierung des Textes sowie Seitennumerierung
– Im Schreibmodus ansprechbare schnelle Routinen zum Verlagern, Kopieren und Löschen von Textteilen sowie zum Einfügen, Auffinden und Ersetzen von Wort- und Satzteilen
– Umschaltbarkeit zu Basic (Belastung des Basic-RAM durch SALOTEX mit nur zwei Byte, Programme bleiben lauffähig)
– Nutzbarkeit von SALOTEX als Screeneditor für Forth, damit Möglichkeit der vollen Kompatibilität zwischen 6502- und U 880-Forth-Rechnern (KC 85/3, ATARI 800 XL bzw. 130 XE).

SALOTEX ist ausschließlich auf der Gerätekombination ATARI 800 XL/ JUNOST 402/GERACORD 6022 (mit Tonbandinterface) mittels eines in Forth verfaßten flexiblen Assemblers/ Disassemblers entworfen worden.

Einige Besonderheiten von SALOTEX:

● In SALOTEX wurde die Lese-/ Schreibroutine des KC 85/3 simuliert. Bereits ohne Interface erhält man vom ATARI 800 XL Recorderaufzeichnungen, die vom KC 85/3 akzeptiert werden. Ein einfaches Interface, das das Ausgangssignal des Recorders auf TTL-Pegel anhebt, gestattet den Datenaustausch auch vom KC 85/3 zum ATARI-Computer.

● In SALOTEX verfaßte Texte können bei Vorhandensein einer V.24-Schnittstelle für den ATARI 800 XL unmittelbar gedruckt werden. Eine reiberroutine und ein Satz der wichtigsten Druckersteuerzeichen für den LX-86 wurde installiert. Zudem können die Texte auf einen KC 85/3 übertragen und mittels einer kurzen Treiberroutine von dessen V.24 zum Druck ausgegeben werden. Verfügt der KC 85/3 über einen TEXOR-Modul, lassen sich die SALOTEX-Texte durch diesen korrigieren und drukken.

Pädagogische Hochschule „Karl Liebknecht" Potsdam, Sektion Marxismus-Leninismus, Am Neuen Palais, Potsdam, 1570

Dr. Petsche

Dienstprogramme für UDOS

Bei der Programmentwicklung für U 880 erzeugen LINK oder PLINK eine Tabelle aller globalen und internen Symbole jedes Objektmoduls. **SYMBOL** gibt diese vollständige Symboltabelle einer Prozedur im Klartext als vierspaltige Tabelle auf SYSLST (unit 3) aus. Die Symboltabelle enthält geordnet nach Objektmodulen jede Assembler-Marke und den dazugehörigen Hexadezimalwert. SYMBOL erleichtert die Fehlersuche in allen Fällen, wo ein Symboldebugger nicht einsetzbar ist, und erweitert die Programmdokumentation.

MORE dient zur bildschirmweisen Betrachtung von Textfiles und kann das Programm PRINT ersetzen.

DIFF vergleicht zwei Textfiles und gibt alle Unterschiede aus. DIFF erkennt auch das Einfügen oder Streichen von Zeilen im Text, vergleicht also auch unterschiedlich lange Texte sinnvoll und geht damit über COMPARE hinaus.

GREP sucht aus Textfiles alle Zeilen heraus, die einen bestimmten Suchbegriff enthalten. Die Suchbegriffe können vollständig festgelegt sein (z. B. Wörter) oder teilweise mehrere Möglichkeiten beinhalten (z. B. beliebige Endungen). Die zusätzliche Ausgabe von Filename, Zeilennummer, Anzahl der gefundenen Einträge u. a. kann gewählt werden.

GREP ist vielseitig einsetzbar für Verwaltung von Katalogen und die Erstellung von Cross-Referenzen, Sachwortregistern und Inhaltsverzeichnissen.

VEB Schiffselektronik „Johannes Warnke" Rostock, Abt. EFB, PF 85, Rostock 5, 2510; Tel. 81 22 20

Penz

dBase III-Dateizugriff und Nutzeroberfläche

Für den erweiterten und multivalenten Einsatz von dBase-Datenspeichern ist es zweckmäßig, einen direkten Datenzugriff zu realisieren, ohne dabei den vollen Systemumfang des Datenbankbetriebssystems einsetzen zu müssen. Es wurde dazu ein Programmpaket Pascal/C/dBase unter dem Betriebssystem MS-DOS entwickelt, das mit folgendem alternativen Leistungsumfang angeboten wird:

– menügeführte Nutzeroberfläche für DBF-Dateien zum Lesen, Schreiben, Aufnehmen und Selektieren von Datensätzen, als lauffähiger geschlossener Modul (COM oder EXE-File)
– Programmbausteine in Quellform für die Einbindung in komplexe Anwendungssysteme in den Versionen für Turbo-Pascal 3.0 und 4.0 oder MS-C 5.0
– Programmgenerator zur Erzeugung von vollständigen und compilierbaren Pascal-Quellmodulen, die jedoch für nutzereigene Erweiterungen offen sind
– Verschiedene Schnittstellen zu CP/M- und SCP-Betriebssystemen, statistische Analyse, Präsentationsgrafik sowie Datentransfer.

Folgende Gebiete bieten sich u. a. für den Einsatz des Programmpaketes an: Datentransfer zwischen verschiedenen Datenverwaltungssystemen, Datentransfer von dBase für numerische Berechnungen und grafische Darstellungen, Operative Datenbankarbeit (UPDATE, LIST, DELETE) mit geringem Datenumfang, Verteilte Verarbeitung der DBF-Dateien auf 8-Bit- und 16-Bit-Technik.

Friedrich-Schiller-Universität Jena, Rechenzentrum, Humboldtstraße 2, Jena, 6900; Tel. 8 20

Scheibe/Wegner

KYRIBASE: dBase versteht Russisch

Das Programm KYRIBASE eröffnet die Möglichkeit, alle Vorteile von dBase III + auch für eine kyrillische bzw. gemischte (lateinisch-kyrillische) Datenbank zu nutzen.

Natürlich hat dBase erstens (1) keinen kyrillischen Zeichensatz und zweitens (2) ist eine Sortierung für kyrillische Textfelder nicht ohne weiteres möglich. Wie werden diese beiden Probleme gelöst?

(1) Ein Textprogramm gibt eine Tastaturbelegung für den kyrillischen Zeichensatz vor. Der Nutzer kann die in lateinischen Lettern geschriebenen „russischen" Feldinhalte lesen und damit auch optisch prüfen (z. B. beim Editieren). Der Ausdruck erfolgt über ein speicherresidentes Programm, für das ein spezieller Druckerfont geschaffen wurde. Ihr grafikfähiger (1) Drucker schreibt nun kyrillische Zeichen.

(2) Für die Sortierung wurde ein Konvertierungsprogramm erstellt. Es werden vom Programm für eine Datenbank automatisch – vom Nutzer zu bestimmende – zusätzliche Felder geschaffen, die die konvertierten Zeichen der originalen „kyrillischen" Felder aufnehmen. Diese Felder kann der Nutzer nun sortieren bzw. indizieren, was sonst nicht möglich wäre.

VEB Mansfeld-Generallieferant Metallurgie, Erzprojekt Leipzig, Abt. JETE, Brandvorwerkstraße 72, Leipzig, 7030; Tel. 3 95 00

Martin

Update-Kopierprogramm UDC

Das Kopierprogramm UDC dient zur Dateisicherung von Festplatte auf Diskette. Es kopiert aus dem aktuellen Verzeichnis nur die Dateien, die auf der Kopiediskette noch nicht gespeichert sind oder dort einen älteren Stand haben (UpDateCopy). Im Programmaufruf können als Parameter mehrere Dateibezeichnungen der Form name.typ unter Verwendung von * bzw.? angegeben werden. Im aktuellen Verzeichnis und auf der Zieldiskette werden die zur Dateibezeichnung gehörenden Dateien mit ihren Datum- und Zeiteintragungen verglichen und danach nur neue und aktuellere Dateien auf die Diskette kopiert. Während des Programmablaufs werden keine Tastatureingaben abgefragt. Das Programm UDC arbeitet als Window. Bildschirmaufbau und Cursorposition werden zum Programmabschluß original restauriert. Damit bietet sich UDC besonders für automatisierte Kopierabläufe an, die z. B. in dBase-Projekten immer wieder benötigt werden.

Dateien mit den Attributen R/O, SYS und HID werden von UDC nicht berücksichtigt.

UDC ist in Turbo-Pascal programmiert und läuft unter MS-DOS und kompatiblen Betriebssystemen. Durch das Anlegen von .BAT-Dateien in allen relevanten Unterverzeichnissen gestaltet sich das aktuelle Sichern von dBase-, Basic- oder Pascal-Programmen sowie TP-Texten und Kalkulationstabellen problemlos. Für diese BAT-Dateien braucht UDC nur in einem dem Betriebssystem durch die PATH-Anwendung bekannten Suchverzeichnis gespeichert sein.

VEB Lufttechnische Anlagen Stahnsdorf, Ruhlsdorfer Straße 23, Stahnsdorf, 1533; Tel. 62 62

Göhrs

Druckprogramm WSPrint

WSPrint ist ein universelles, menügesteuertes Druckprogramm in Ergänzung zu TP (Textprozessor). WSPrint wurde in Turbo-Pascal implementiert und ist unter SCP 1700 am A 7100 verfügbar.

In WSPrint wurde gegenüber TP ein erweiterter Steuerzeichen- und Punktkommandovorrat realisiert. So sind beispielsweise die Grafikzeichen der EPSON-Drucker verfügbar, der Zeilenabstand läßt sich in 1/48"-Schritten variieren. Textdateien können auf dem Bildschirm in 10 Geschwindigkeitsstufen angezeigt werden, wobei darstellbare Steuerzeichen interpretiert werden, die Seitenformatierung wird dabei durchgeführt. Dadurch ist eine erhebliche Papiereinsparung bei der Textverarbeitung möglich.

WSPrint erzeugt auf allen implementierten Druckertypen (EPSON LX 86, FX 1000, robotron K 6313/14, K 6316 (Schalterdrucker), SD 1152 (deutsches und internationales Typenrad)) annähernd gleiche Druckeffekte. Nicht darstellbare Druckmodi (z. B. NLQ und Hoch-/Tiefstellen beim LX 86) werden in anderer geeigneter Form (z. B. Doppeldruck) dargestellt. Im Zeichensatz des jeweiligen Druckertyps nicht vorhandene Symbole werden improvisiert (beispielsweise] als / beim 1152 mit deutschem Typenrad) oder als Leerzeichen gedruckt (einige Grafikzeichen). WSPrint ermöglicht eine betriebsinterne Standardisierung des Steuerzeichenumfangs in Textdateien, da der Druck nicht mehr an eine druckerspezifisch generierte TP-Version gebunden ist.

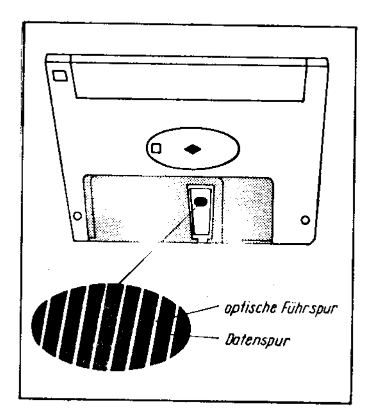
Technische Universität Dresden, Sektion Fertigungstechnik und Werkzeugmaschinen, Bereich Montage und Fügetechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4 57 93 34

Heidenbluth

Erster optischer 1-KBit-Chip

Den weltweit ersten optischen 1-KBit-Speicher hat das japanische Elektronikunternehmen NEC entwickelt. Dieser Hochgeschwindigkeitsspeicher mit 1024 optischen Elementen auf einem ein Quadratmillimeter großen Stück sei ein Schritt in Richtung eines 1-MBit-Chips auf der Basis von Galliumarsenid, der in einigen Jahren zur Verfügung stehen könnte.
Der neue Chip soll in der Lage sein, eine Vielzahl optischer Signale parallel zu verarbeiten – und das viel schneller als konventionelle elektronische Schaltkreise. Nach Angaben des Unternehmens könnte der neue Speicher den Weg zu optischen Computern eröffnen.
ADN

Floptical Disk mit mehr als 20 MByte



Anläßlich der Comdex-Messe im November 1988 in Las Vegas erregte das junge US-Unternehmen Insite Peripherals mit seinem opto-magnetischen Floppy-Disk-Laufwerk *Insite Model 1325* besondere Aufmerksamkeit. Das Interessante an dem Laufwerk ist, daß es mit besonders nachbehandelten, aber ansonsten handelsüblichen 3,5-Zoll-Disketten Speicherkapazitäten bis zu 20,8 MByte formatiert (25 MByte unformatiert) erlaubt (siehe auch MP 2/89, S. 61). Möglich ist dies durch optisch auswertbare 4,5 µm breite Führspuren, die mittels Laserstrahl konzentrisch bei Insite Peripherals auf die Aufzeichnungsoberfläche geschrieben werden (siehe Bild). Das Licht einer Leuchtdiode (LED) wird durch eine Öffnung im Schreib-/Lese-Kopf des Laufwerkes auf die optische Führspur geworfen, dort reflektiert und über Fotozellen ausgewertet. Damit wird eine besonders hohe Positioniergenauigkeit erreicht. Sie erlaubt es nun, statt der bisher maximal möglichen 48 bis 135 Spuren 1250 Spuren mit 15,24 µm Breite aufzuzeichnen.
Die mittlere Zugriffsgeschwindigkeit wird mit 65 ms angegeben.
Zum Anschluß an PC XT/AT- und PS/2-Modelle ist das Laufwerk mit der SCSI-Schnittstelle ausgestattet. Es soll Mitte des Jahres auf den Markt kommen; die Firmen Kodak, Verbatim und Xidex werden die Herstellung der Disketten übernehmen. *MP*

MCI Carat IV

Den nach eigenen Angaben mit 33 MHz schnellsten 80386-Mikrocomputer hat die Firma MCI auf der Electronica '88 gezeigt. Gedacht ist der als Tower ausgeführte Carat IV für CAD/CAM-Anwendungen und dementsprechend mit schneller Grafik sowie Festplatten hoher Kapazität ausgestattet. Nachrüstbar ist ein Weitek-Koprozessor. Betriebssysteme sind wahlweise DOS oder SCO Xenix 386. *MP*

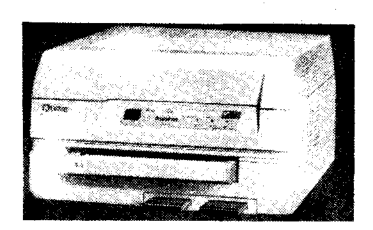
32-Bit-Europrozessor

Neun europäische Computer- und Schaltkreishersteller überlegen gegenwärtig die gemeinsame Entwicklung eines eigenen Mikroprozessors, um die Abhängigkeit speziell von US-amerikanischen Firmen zu verringern. Siemens und Philips, die Halbleiterfirmen Alcatel, Plessey und SGS Thomson Microelectronics sowie die Computerfirmen Bull, International Computer Ltd., Nixdorf und Olivetti diskutieren über einen sogenannten Europrozessor, der in einer 32-Bit-Version innerhalb von drei Jahren produktionsreif sein soll. Genaue Details des Europrozessor-Programms sind noch nicht bekannt, wahrscheinlich ist aber ein RISC-Prozessor in CMOS-Technologie. Ausgangspunkte könnten Hyperstone, ein von dem Ingenieurbüro Otto Müller in Konstanz entwickelter 32-Bit-CMOS-Prozessor mit einer Leistung von 20 MIPS und ARM, ein RISC-Prozessor der britischen Firma Acorn Computer Ltd., die von Olivetti kontrolliert wird, sein. Angestrebt wird ein Prozessor, der UNIX-kompatibel und paßfähig zu den Systembusstrukturen der 80386- und 68020/68030-Prozessoren sein soll. *MP*

Laptop mit Farbdisplay

Den nach eigenen Angaben ersten Laptop-Computer mit Farbmonitor stellte jetzt die japanische Firma Hitachi vor. Es handelt sich dabei um ein 6,3-Zoll-Flüssigkristalldisplay mit einer CGA-Auflösung von 640 × 200 Bildpunkten; im Textmodus sind 25 Linien mit 40 oder 80 Zeichen darstellbar. Weitere Merkmale des HL 400C sind ein 80286-Prozessor mit 12 MHz, 1 MByte RAM (der sich erweitern läßt), je eine parallele, serielle und RGB-Schnittstelle, zwei Steckplätze, eine 3,5-Zoll-Floppy und eine 20-MByte-Festplatte. Die Masse des Laptops beträgt 7,3 kg. *MP*

Neue CrystalPrint-Familie von Qume



Die Qume GmbH stellte Ende 1988 eine völlig neue Seitendrucker-Familie vor. Bei den Geräten wird die von der Firma Casio entwickelte LCS- (Liquid Crystal Shutter-) Technologie verwendet. Das heißt, die Drucker arbeiten nach dem gleichen Grundprinzip wie die meisten LED- oder Laserdrucker, stellen also praktisch eine Kombination aus Fotokopierer, Laserdrucker und Matrixdrucker dar: Eine lichtempfindliche Trommel wird punktweise belichtet und das so entstandene Bild mit Hilfe von Toner, Hitze und Druck auf Papier übertragen. Der Unterschied liegt in der Lichtquelle. Während andere Seitendrucker überwiegend den Laserstrahl oder LED-Schienen als Lichtquelle einsetzen, verwendet die LCS-Technologie eine einfache Halogen-Stablampe. Zwischen dieser Lampe und der Bildtrommel befinden sich ein Flüssigkristall-Lineal (Liquid Crystal Shutter) mit nebeneinander angeordneten Flüssigkristallen und ein Linsenfeld. Die 300 Flüssigkristalle pro Zoll können durch Anlegen einer Spannung in ihrer Ausrichtung geändert und so je nach Vorgabe des Controllers lichtdurchlässig gemacht werden. Das Licht der Stablampe fällt entsprechend nur durch die durchlässigen Kristalle und wird anschließend über das Linsenfeld auf die Trommel fokussiert. Der weitere Druckablauf – Entwickeln und Fixieren – entspricht dem der anderen Seitendrucker.
CrystalPrint wird von Qume in zwei Modellen mit den Zusatzbezeichnungen „WP" und „Series II" angeboten. Die Variante „WP" ist primär als Ersatz von Typenraddruckern und höherwertigen Matrixdruckern gedacht. Der CrystalPrint Series II soll seinen Einsatz als DTP- und Grafikdrucker finden und verfügt daher serienmäßig über eine HP Laserjet Serie II-Emulation mit bis zu 1,5 MByte Speicher. Die Druckgeschwindigkeit wird mit sechs Seiten pro Minute angegeben.
MP

BRD-Forscher entwickelten superschnellen Siliziumchip

Zwei Bochumer Forschern ist erstmals der Beweis gelungen, daß die Grenzen bei Silizium als einfachem und billigem Ausgangsmaterial für die Halbleitertechnik noch längst nicht ausgenutzt sind. Mit ihrer Neuentwicklung, einem superschnellen Bipolartransistor auf Siliziumbasis, können die Experten bereits in Geschwindigkeitsbereiche jenseits der „Schallmauer" von 10 Gigabit pro Sekunde vordringen, die bisher für Siliziumchips als unerreichbar galten. Aus diesem Grund seien gerade für die nächsten Computergenerationen vielerorts Forschungen im Gange, die über Silizium hinaus gingen und den komplizierten und teuren Verbindungshalbleitern auf der Basis von Galliumarsenid gelten. Daß Geschwindigkeiten von 20 bis 30 Gigabit pro Sekunde tatsächlich auch mit Silizium erreicht werden können, sei inzwischen mit dem ersten „echten" Si/SiGe-Heterobipolartransistor (HTB) bewiesen worden. Mit diesem neuen Transistor aus Bochum würden jetzt auch die vorteilhaften Eigenschaften der Verbindungshalbleiter mit Silizium als Grundmaterial erreicht.
ADN

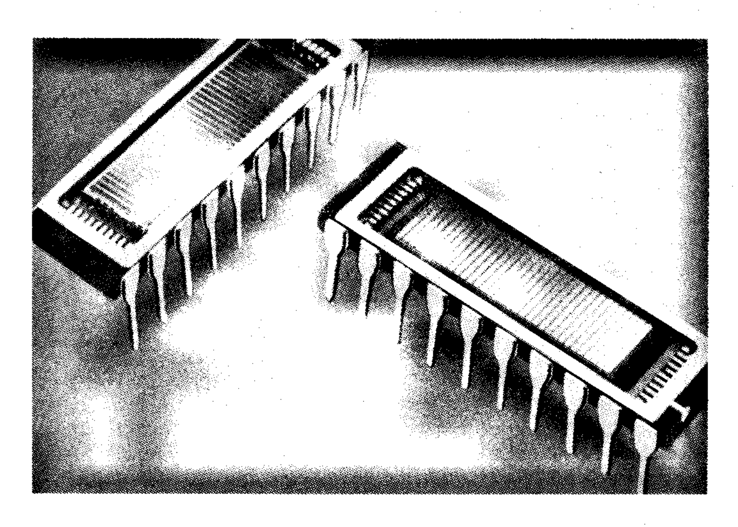
Künftig 32-Bit-RISC-Prozessoren von Siemens

Die Siemens AG hat im Februar eine langfristige Vereinbarung mit der kalifornischen MIPS Computer Systems, Sunnyvale, getroffen. Danach erhält Siemens die Möglichkeit, die 32-Bit-RISC-Prozessorfamilien von MIPS einschließlich der Folgegenerationen zu produzieren und mit eigener Entwicklung aus den Prozessoren Derivate wie Microcontroller und Prozessorkerne abzuleiten. Das Abkommen bezieht sich zugleich auf die dazugehörige Software wie Betriebssystem, Compiler und Support-Programme. Die ersten Prozessoren sollen im IV. Quartal dieses Jahres geliefert werden. Die japanische NEC Corporation hat mit MIPS zum gleichen Zeitpunkt ein ähnliches Abkommen geschlossen.
„Im Zeichen globaler Märkte sind heutzutage neue Strategien und Allianzen unter starken Partnern ein Muß, nicht nur im Bereich der Systeme, sondern auch bei Halbleitern", erklärte Dr. Horst Fischer, Mitglied der Leitung des Geschäftsführenden Bereichs der Siemens AG. MIPS sei ein starker Partner, der gemeinsam mit seinen Halbleiterherstellern in den USA im letzten Quartal 1988 mehr RISC-Prozessoren verkauft hat als alle anderen Anbieter zusammen. Bedeutende Systemanbieter wie DEC haben für ihre neuen Generationen von Workstations oder UNIX-Systemen bereits diese RISC-Architektur übernommen.
Siemens hat sich für die RISC-Prozessoren von MIPS entschieden, weil sie mit derzeit 20 MIPS (Millionen Instruktionen pro Sekunde) durchschnittlicher Systemleistung zu den weltweit schnellsten zählen und weil für die Anwendung dieser Prozessoren das mit Abstand breiteste Software-Angebot vorliegt, insbesondere optimierte Compiler für höhere Programmiersprachen wie auch leistungsfähige Entwicklungssysteme. Darüber hinaus ist diese RISC-Prozessorfamilie die einzige am Weltmarkt mit Pinkompatibilität unabhängig vom Hersteller. *MP*

Intel-kompatible Prozessoren von NEC

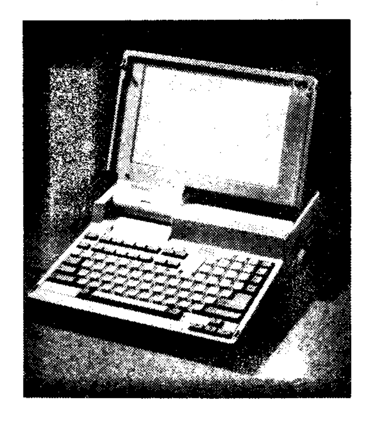
Der neueste 80386-kompatible Prozessor in CMOS-Technologie von NEC ist der V70. Mit einer Taktfrequenz von 20 MHz leistet er 6,6 MIPS – andere Angaben gehen bis zu 45 MHz und 15 MIPS. Damit ist er schneller als der 80386 von Intel, der bei 25 MHz eine Rechnerleistung von 5 MIPS liefert. Die Transferrate des V70 beträgt 40 MByte pro Sekunde.
Die intel-kompatible V-Reihe mit 16-Bit-Verarbeitungsbreite beginnt beim V20 (8088-kompatibel) und geht über die Typen V30, V33 und V50 (8086-kompatibel) bis zum V40 (80186-kompatibel). (Der V50 ist allerdings nur mit dem 8086-Modus des 80286 kompatibel.) Die Prozessoren mit 32-Bit-Verarbeitungsbreite sind der V60 mit einem 16-Bit-Datenbus (ähnlich dem Intel 80386SX) und der V70.
Wegen ihres geringen Stromverbrauchs werden die Prozessoren der V-Serie bevorzugt in Laptops eingesetzt. *MP*

16-MBit-DRAM von Matsushita



Im Semiconductor Research Laboratory von Matsushita wurde der Prototyp eines 16-MBit-DRAMs entwickelt. Er wurde in einer 0,5-μm-Technologie gefertigt und bringt die bemerkenswert geringe Zugriffszeit von 65 ns. Der Chip, der auf einer Fläche von 94 mm^2 rund 35 Millionen Transistoren enthält, ist nur 40 % größer als der 4-MBit-DRAM, den Matsushita auf der Electronica '88 im November in München vorstellte. *MP*

Neue Laptop-Computerreihe von Sharp



Auf dem Markt der Laptops verfolgen die Hersteller offensichtlich unterschiedliche Ziele. Einige Hersteller rüsten ihre Geräte technisch immer weiter auf (s. MP 1/89, 4. Umschlagseite). Ein etwas anderes Konzept verfolgt Sharp Electronics. Das Konzept hieß von Anfang an, möglichst nur „reinrassige" Laptops zu bauen, das heißt, es sollten nur besonders leichte, handliche und vor allem netzunabhängige Geräte angeboten werden. Diese Grundsätze verwirklicht Sharp auch bei seiner neuen PC-46XX-Serie.
Der PC-4602 enthält zwei 3½-Zoll-Laufwerke und der PC-4641 ein 3½-Zoll-Laufwerk sowie ein 40-MByte-Festplattenlaufwerk. Sharp präsentiert damit den nach eigenen Angaben ersten netzunabhängigen XT-kompatiblen Laptop mit einer derart hohen Speicherkapazität. Die Zugriffszeit bei der Festplatte (PC-4641) liegt bei 55 ms. Die interne Speicherkapazität beträgt bei beiden Modellen in der Standardausstattung 640 KByte. Für höhere Ansprüche kann der RAM optional auf 1,6 MByte ausgebaut werden. Als Prozessor wird der Intel-Typ 80188 mit 10 MHz eingesetzt.

Das Display ist ein hintergrundbeleuchtetes und äußerst kontrastreiches *Reverse-Reverse-Supertwisted-LCD*. Text und Grafiken werden mit einer Auflösung von 640 Pixel (horizontal) × 400 Pixel (vertikal) dargestellt. Bei einer Bildauflösung von 640 × 200 werden für Grafiken die vertikalen Pixel verdoppelt (Double-Scan). Eine Bildauflösung von 320 × 200 läßt eine vertikale und horizontale Verdopplung der Pixel zu (2 × 2). Durch Ändern der beleuchteten Pixel aus diesen vier Pixeln lassen sich vier Grauskalen zur Andeutung unterschiedlicher Farben erzielen. Für den Anschluß von Drucker, Modem, Maus usw. steht neben der Centronics- und der RS-232 C-Schnittstelle (9-pin) ein Anschluß für ein externes 5,25-Zoll-Laufwerk zur Verfügung. Eine zweite serielle Schnittstelle ist für den gleichzeitigen Betrieb von seriellem Drucker und Maus vorhanden. Ebenso ist ein Farb-Monochrom-Bildschirm-Adapter zum Anschluß eines externen Farbbildschirmes optional erhältlich. Inklusive Batterie, jedoch ohne das Netzteil, wiegt der PC-4602 weniger als 5,5 kg und der PC-4641 weniger als 6,1 kg. Als Software ist im Lieferumfang das Betriebssystem DOS 3.3 sowie GW-Basic 3.22 enthalten. *MP*

Transistoren und Dioden aus Polyacetylen

Im Cavendish-Laboratorium der Universität in Cambridge wird an der Entwicklung einer neuen Generation von Halbleitern gearbeitet, die aus organischen Materialien bestehen und künftig im Fernsprechverkehr und in anderen Elektronikerzeugnissen angewendet werden könnten. Dabei gelang es bereits, Transistoren und Dioden aus Polyacetylen herzustellen. Durch den Einsatz eines chemischen Verfahrens, das an der britischen Universität Durham entwickelt worden ist, gelang es, extrem dünne Filme aus Polyacetylen herzustellen und auf die Chip-Komponenten zu übertragen. Die Filme reagierten klar und verläßlich auf elektrische Signale und wandelten ihre Farbe mit der Veränderung der Spannung, was bisher bei Polyacetylen nicht der Fall war. Mit den bisherigen Modellen steht man erst am Anfang der Entwicklung. Die Modelle sind noch zu umfangreich und arbeiten zu langsam, um sie schon zum Einsatz bringen zu können.
Quelle: Blick durch die Wirtschaft vom 13. 09. 1988 *Fa*

Hochleistungschip von NEC

Im Rahmen ihres nationalen Supercomputer-Forschungs- und Entwicklungsprojektes entwickelte die japanische Firma Computer Engineering Division der NEC Corp. einen Hochleistungsmultichip, auf dem 21 GaAs- und 47 bipolare Siliziumchips auf einer Grundfläche von 100 × 100 mm^2 angeordnet sind. Die elektrischen Verbindungen zwischen den Bauelementen befinden sich unter dieser Fläche in mehreren Ebenen. Die einzelnen Verbindungsebenen werden durch jeweils 20 μm dicke Polyimidfilme voneinander isoliert. Die Leiterbahnen liegen in der Größenordnung von 25 μm.
Als Grundlage wurde kein Siliziumwafer, sondern ein Saphirwafer genutzt, dessen geringere Oberflächenrauigkeit dünnere Leiterbahnen möglich macht. Als mechanischer Träger der gesamten Anordnung wird ein Aluminiumoxidsubstrat benutzt. Die Gesamtverlustleistung des Prototyp-Multichips beträgt 150 W. Um diese Leistung abzuführen, wird der Baustein mit flüssigen Fluorkohlenstoffverbindungen gekühlt, so daß die Betriebstemperatur nicht über 80 Grad steigt. Die kompakte Packungsdichte und die kurzen Verbindungen ermöglichen Verzögerungen von höchstens 6,5 ps/mm.
Quelle: Electronics. – New York (1988) 11 *Wi*

GaAs-Chip mit Schaltfrequenz von 3,5 ps

Ein von der Stanford Universität entwickelter GaAs-Chip weist Schaltfrequenzen von 3,5 ps bei Raumtemperaturen auf und könnte damit die im 35-Pikosekundentakt arbeitenden Meßdioden ersetzen. Dieser Schalttakt entspricht einem Frequenzbereich von etwa 100 GHz.
Die Herstellung des Chips erfolgte auf der Grundlage der Verwendung eines Lasersystems, das mit einem äußerst schnell arbeitenden Blendensystem ausgerüstet wurde.
Die Schaltzeiten von 3,5 ps liegen in der Nähe der theoretischen Werte für sogenannte Josephson-Schaltelemente. Als nächsten Entwicklungsschritt will man Schaltfrequenzen im 1-ps-Bereich erreichen.
Quelle: Blick durch die Wirtschaft von 22. 11. 1988 *Fa*

Weitere Rechnerprojekte mit Galliumarsenid-Schaltkreisen

Nachdem die Firma Cray ihren ersten Computer auf der Basis von Galliumarsenid-Bauelementen für das Jahr 1989 angekündigt hat (siehe MP 9/1988), wird vermutet, daß die Firma IBM ihre mittleren und großen Computer ab 1995 mit derartigen Bauelementen ausrüstet. Bis zu diesem Zeitpunkt wird IBM die Galliumarsenidschaltkreise bereits in Kommunikationskanälen, Cache-Speichern und -Controllern einsetzen. Die Technologie für die Galliumarsenidschaltkreise will IBM von der Firma Rockwell International übernehmen, die bereits seit Mitte der siebziger Jahre führend auf diesem Gebiet ist.
Die Galliumarsenidlogikschaltkreise für den Rechner Cray-3 sollen von der Firma Gigabit Logic Inc. entwickelt worden sein. Durch die Verringerung der internen Verbindungen wirkt sich die höhere Arbeitsgeschwindigkeit der GaAs-Bauelemente voll aus. Der Rechner mit 16 Prozessoren und 4 Gigabyte Hauptspeicher soll eine Leistung von 16 GFLOPS (Milliarden Gleitkommaoperationen pro Sekunde) erreichen (Vergleich Cray-2 1,8 GMFLOPS)
Quelle: Electronics. – New York 61 (1988) 12
Blick durch die Wirtschaft vom 21. 12. 1988 *Wi*

Vorbehalte hinsichtlich des Nutzens von Hochtemperatursupraleitern

Hochtemperatursupraleiter bringen nach Ansicht von Texas Instruments in absehbarer Zeit keine Vorteile in der praktischen Anwendung gegenüber konventionellen elektronischen Schaltungen und Leiterwerkstoffen. Die niedrige Schaltenergie würde durch den Energieaufwand für die Kühlung kompensiert. Ein weiterer Nachteil ist die Energiezunahme bei der Verkleinerung der Schaltelemente, die somit der Miniaturisierung entgegensteht. Die langsamen Schaltvorgänge in peripheren Funktionen bringen im Gesamtsystem die extrem kurzen Umschaltzeiten der supraleitenden Schaltung nicht zur Wirkung. Die Vertreter von Texas Instruments sind außerdem der Auffassung, daß mit Kupfer, das auf die Temperatur der Hochtemperatursupraleiter gekühlt wird, dieselbe Leitfähigkeit wie bei den Supraleitwerkstoffen erzielt wird.
Quelle: Eildienst. – Berlin (1988) 252. – S. 5 *Wi*

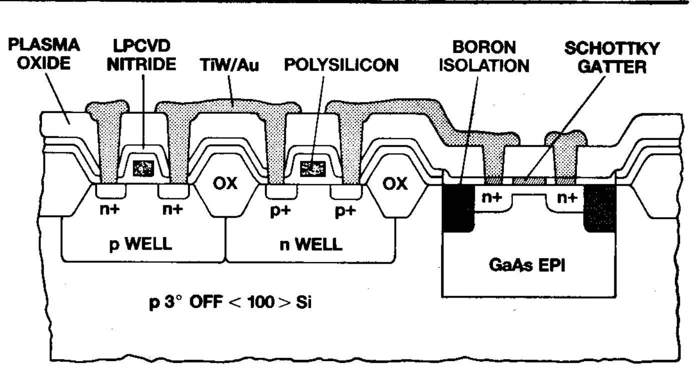
Projekt „5. Rechnergeneration in Japan" mit Verzögerung

Die Bereitstellung eines Prototyps für einen Rechner der Fünften Generation soll auf Schwierigkeiten stoßen. In den ersten sieben Jahren wurden die Grundlagen der Künstlichen Intelligenz und der Verarbeitung menschlichen Wissens geschaffen. Die Umsetzung dieser Erkenntnisse in einen Computer mit Parallelverarbeitungsmöglichkeiten und die vorgesehenen Programmierverfahren waren von ausländischen Computerwissenschaftlern für unreal angesehen worden. Wissenschaftler aus Großbritannien, den USA und Frankreich sollen nun einbezogen werden, um mit deren Unterstützung das Projekt zum Abschluß zu bringen.
Zwischenzeitlich wurde durch das japanische Ministerium für internationalen Handel und Industrie ein Programm geschaffen, das einen neuralen Computer zum Ziel hat.
Quelle: Eildienst. – Berlin (1988) 256. – S. 4 *Fa*

Electronica '88

Vom 8. bis 12. November 1988 fand in München wiederum die Electronica statt, verbunden mit Fachtagungen, darunter dem 13. Internationalen Kongreß Mikroelektronik. Die Fachmesse für Bauelemente und Baugruppen der Elektronik war auf 105000 m^2 Hallenfläche übersichtlich in die drei Angebotsbereiche Elektronik-Komponenten, Elektromechanik-Produkte sowie Ausstattungen für Entwicklung und Qualitätssicherung gegliedert.
Die Breite und das werbewirksame Niveau der Darbietung machten deutlich, daß stabile Kooperationsbeziehungen sowohl zwischen den Anbietern als auch hin zu den Anwendern eine wesentliche Quelle sind für die schnellstmögliche Realisierung neuer Produktideen und die Wettbewerbsfähigkeit ganzer Industriezweige. Pressemitteilungen sprachen von der existentiellen Bedeutung des Zuliefersektors für hohe Wachstumsraten.
Der 13. Internationale Kongreß Mikroelektronik vermittelte in Fachsitzungen sowohl Informationen über neue Produkte als auch zu Trends und Forschungsprojekten.
In der Sektion Galliumarsenidtechnik beispielsweise wurde die monolithische Mikrowellen-Schaltkreistechnologie als ausgereiftes Verfahren der Gegenwart gewertet. GaAs hat bei nahezu allen Anwendungen physikalisch begründete Vorteile; es ermöglicht erst die Höchstfrequenztechnik über 30 Gigahertz. Die Grenzen seiner Anwendung sind gegenwärtig ausschließlich den Herstellungskosten der Bauelemente geschuldet (Ausgangsmaterial 10mal teurer, Ausbaute geringer, Integrationsniveau 1000fach geringer).
Die Fachsitzung „Kreative Mikroelektronik" stellte die Entwicklung anwendungsspezifischer Schaltkreise (ASICs) in den Vordergrund. Darin wird eine universelle Methode der Höchstintegration gesehen, deren Anwendungsfeld 1990 bis 25% des Marktes an integrierten Schaltkreisen erreichen wird. Neue Produkte zeichnen sich durch hohe Komplexität, überwiegende Anwendung der 1-Mikrometer-CMOS-Technologie, 100 Tausend Gatter bei Gate-Arrays, 200 Tausend bei Standardzellen und Zugriffszeiten von 0,5 Nanosekunden aus. Zunehmend findet die analog-digitale Systemintegration (BICMOS-Technologie) Anwendung. Für 1990 wird ein Anteil von 40% gemischt digital-analogen ASICs erwartet.
Für eine allgemeine Wertung der zur Electronica '88 vorgestellten Produkte sollte besonders hervorgehoben werden, daß sich die CMOS-Technologie bei unipolaren Schaltkreisen vollständig durchgesetzt hat und alle führenden Schaltkreishersteller das mit dem Innovationstreiber Speicher erreichte höhere Technologieniveau gleichzeitig auf verbesserte Mikroprozessoren, Signalprozessoren, ASICs u. a. übertragen.
Dazu im folgenden einige Beispiele:
● Vom 1-MBit-DRAM wurden 1988 weltweit über 225 Millionen Stück hergestellt (und noch weit mehr benötigt). Auf der Electronica wurden Muster von 4-MBit-DRAMs gezeigt, deren Produktion 1989 beginnen soll, ohne daß zunächst große Stückzahlen abgesetzt werden.
Das Eureka-Projekt Jessi, das sich derzeit in der Definitionsphase befindet, hat das Ziel, die europäischen Unternehmen bis Mitte der neunziger Jahre technologisch in die Lage zu versetzen, 64-MBit-DRAMs herzustellen: Chips also, die über 4000 Schreibmaschinenseiten (nahezu 8,4 Millionen Zeichen) speichern können.
● Die Firma VLSI-Technology gab den Produktionsanlauf des ersten Abschnitts einer neuen Hochleistungsfabrikation für Bauelemente auf 6-Zoll-Wafer-Basis bekannt, für die im März 1987 der Grundstein gelegt wurde. Für diesen Abschnitt werden 150–200 Millionen Dollar Umsatz pro Jahr erwartet. Die Gesamtanlage soll 230000 Quadratmeter Produktionsfläche umfassen, davon 37000 m^2 in Reinraumklasse 1. Es sollen Schaltkreise mit bis zu 0,25-µm-Strukturen gefertigt werden.
● Texas Instruments wirbt für die Anwendung seiner direktschreibenden Elektronenstrahlanlage zur Herstellung kunspezifischer Schaltkreise in HCMOS-Technologie, mit 1,8- und 3-Mikrometer-Strukturbreiten sowie mit 300 bis 2500 nutzbaren Gattern. Damit will die Firma den Zeitraum vom Kundenentwurf bis zur Lieferung montierter und getesteter Prototypen auf etwa 1 Woche reduzieren.
● 32-Bit-Mikroprozessoren herkömmlicher Bauart werden in vielfältigen und leistungsgesteigerten Varianten angeboten: höhere Geschwindigkeiten, höhere Integration, mehr interne Speicher.
Intensiv gearbeitet wird an Prozessoren mit reduziertem Befehlssatz (RISC-Mikroprozessoren). Motorola beispielsweise stellt mit der Familie M 88000 ein komplettes System vor:
- CPU MC 88100
- Daten- und Instruktionseinheit
- Integereinheit für arithmetische, logische Bitfeldoperationen
- Fließkommaeinheit
- Cache/Speicherverwaltungsschaltkreis.

Diese neue RISC-Prozessorfamilie erweitert Motorolas Angebot an Mikroprozessoren nach oben in höhere Leistungsbereiche und wird neue Märkte für Mikroprozessoren erschließen. Es soll das erste vollständige Konzept auf dem Markt sein, mit vielen heute schon verfügbaren Entwicklungshilfsmitteln ausgestattet.
● In diesem Zusammenhang ist der digitale Signalprozessor TMS 320 C 30 von Texas Instruments zu nennen. Er umfaßt 695000 Transistoren in 1-Mikrometer-Technologie, hat eine Befehlszykluszeit von 60 Nanosekunden, führt 33 Millionen Gleitkommaoperationen pro Sekunde aus und baut auf einer anwendungsspezifischen RISC-Architektur auf. Typische Anwendungen sind Workstations (Vektorberechnungen bei dreidimensionalen Bildern), Radarapplikationen, Berechnungen von Hochgeschwindigkeitsdatenprozessoren in Rechnern. Der auf dem Chip integrierte DMA-Controller kann im gesamten Speicherbereich des Prozessors Daten austauschen, ohne die Zentraleinheit zu unterbrechen.



● Texas Instruments informierte über Forschungsergebnisse und -muster, mit denen Silizium- und Galliumarsenid-Transistoren zu einem monolithischen Schaltkreis integriert werden. Ziel der Arbeit ist eine Prozeßtechnologie für eine auf Silizium basierende Serienproduktion. Dazu wurden Siliziumbauelemente auf Standard-Silizium-Wafern hergestellt, auf diese selektiv dünne, planare GaAs-Inseln aufgebracht, darauf die GaAs-Strukturen produziert und schließlich mit Standardtechnologien die Metallverbindung aller Komponenten vorgenommen (siehe Bild). Auf diese Weise wurde begonnen, die Integration der eigentlich unverträglichen hochdichten Siliziumschaltungen und extrem schnellen GaAs-Komponenten an einfachen Testschaltungen zu vollziehen und Erfahrungen für die Realisierung dieser Technologie unter Produktionsbedingungen zu schaffen. Als Anwendungen zeichnen sich die Herstellung von Rechnern und EDV-gestützten Systemen ab, die um vieles schneller und zuverlässiger sind als heutige Anlagen, die Darstellung und Grafik mit höherer Auflösung erlauben und im Vergleich zu ihren Vorgängern durchaus auch kleiner, leichter, kostengünstiger und energiesparender sein könnten. Zum Beispiel ist es möglich, Computersysteme mit 1-MBit-Silizium-Speichern mit einer GaAs-Hochleistungssteuerlogik zu erweitern. Denkbar wäre auch ein hochdichter Silizium-Mikroprozessor mit einem integrierten GaAs-Registerfile oder einem Cache-Speicher. Logik- und Speicherchips mit optischen GaAs-on-Board-Komponenten können Vorteile bieten. Ein GaAs-Laser überträgt Daten mit 5 bis 10 GBaud, also wesentlich schneller als heute über Kabel realisierbar.
● Als größte Renner der nächsten Jahre in der Leistungselektronik werden die intelligenten Leistungsschaltkreise gewertet, bei denen sowohl die Logik als auch die Leistungsstufe auf ein und demselben Chip vereint sind, und zwar immer häufiger mit gemischten (bipolaren und MOS-) Technologien. Ihnen wird ein Wachstum von rund 50 Prozent pro Jahr vorhergesagt. Das Einsatzgebiet für diese sogenannte Smart-Power ist die Schnittstelle zwischen dem Computer bzw. den Mikroprozessoren auf der einen Seite und peripheren Leistungsverbrauchern auf der anderen. Letztlich entspringt diese Entwicklung der Notwendigkeit für Systemhersteller, elektromechanische Baugruppen durch Elektronik zu ersetzen, um Zuverlässigkeit und Funktionalität zu verbessern, neue Anwendungen zu erschließen, Platz und Energie zu sparen und die Kosten für die Systemfertigung zu senken.

Ha

6. Problemseminar Programmierung von Rechenanlagen des SKR

Von der Kammer der Technik, Bezirksvorstand Neubrandenburg, und dem Informatik-Zentrum des Hochschulwesens an der TU Dresden wurde vom 29.11.–2.12.88 in Neubrandenburg das 6. Problemseminar „Programmierung von Rechenanlagen des SKR" mit 137 Teilnehmern durchgeführt. Das Tagungsprogramm umfaßte 22 Vorträge zu Problemen des Einsatzes von Rechenanlagen des SKR mit dem universellen Echtzeitbetriebssystem OS-RW (MOOS-1600). Schwerpunkte der Tagung waren vor allem der Einsatz und die Nutzung der Standardsoftware und die Einbindung in Rechnernetze auf der Basis der SMnet (SKRnet)-Architektur.
Dr. Galler (FSU Jena) berichtete über die Einsatzerfahrungen mit der Elektronika-79 (UdSSR), die mit einer Befehlsausführungszeit für Register-Register-Befehle von 0,24 µs die leistungsfähigste 16-Bit-SKR-Anlage ist. Bei eingeschaltetem Cache-Speicher ergibt sich gegenüber der SM4-10 für Real*4-Daten eine 6fache und für Real*8-Daten eine 50fache Leistungssteigerung. Weitere Vorteile sind die Trennung von Daten- und Befehlsadreßraum und die 100-MByte-Plattenspeicher EC 5067. Kritisch wurden auch Erfahrungen der technischen Zuverlässigkeit und Verfügbarkeit ausgewertet.
Domsche (VEB Behälterglas Bernsdorf) und *Hachenberger* (VEB Braunkohlekombinat Bitterfeld) werteten ihre Erfahrungen der Nutzung des Betriebssystems OS-RW auf einer SM14-20 mit 2-MByte-Hauptspeicher und ausgebauter Konfiguration im extremen Multiuserbetrieb mit 10 bis 14 Nutzern aus. Es wurden Hinweise zur Erhöhung des Datendurchsatzes und Verbesserung der Time-Sharing-Eigenschaften gegeben.
Dr. Bock (TH Wismar) diskutierte Probleme der grafischen Datenverarbeitung auf Anlagen SM4-ARM mit umfangreicher Grafikperipherie. Im

Vortrag wurde herausgearbeitet, daß gegenwärtig nur das grafische Kernsystem GKS zum Einsatz kommen kann, da IGES nicht für die Geometriemodellierung geeignet ist und STEP erst in etwa 4 bis 5 Jahren anwendbar sein wird.
Hoffmann (Deutsche Staatsbibliothek Berlin) stellte in einem sehr guten Übersichtsbeitrag das interaktive relationale Datenbanksystem IDAS vor, das gegenwärtig nur unter UNIX-kompatiblen Betriebssystemen, wie MUTOS-1600 oder DEMOS, eingesetzt werden kann. Der Vorteil von IDAS ist u. a. die Anwendbarkeit variabel langer Felder. Programminterfaces existieren für C, Fortran, Pascal und Cobol. IDAS-Versionen gibt es unter ESER/PSU, ESER/VMX und MUTOS-1800.
Dr. Utke (TU Dresden/Informatikzentrum) und *Papenfuß* (FSU Jena) stellten die SMnet-Software für das OS-RW vor und diskutierten Anwendungsprobleme bzw. erste Einsatzerfahrungen. Schwerpunkte waren u. a. die Kopplung zu SKRnet des K1840 und die Einbindung von Personalcomputern unter DCP-1700.
In einer abschließenden Problemdiskussion wurden Probleme des Einsatzes von Datenbanksystemen, die Integration von PC-Netzen, Probleme des Einsatzes von Terminalemulatoren und Fragen der Erhöhung der Zuverlässigkeit der Wechselplattenspeicher erörtert.

Prof. Dr. Thomas Horn

UNIX-Börse

Die Entwickler- und Anwendergemeinschaft UNIX-kompatibler Software (UNIX-EAG) führte vom 7. bis 11. November 1988 im VEB Leitzentrum für Anwendungsforschung ihre erste Softwarebörse durch. Die angebotene Software wurde auf verschiedenster Rechentechnik präsentiert und ein Teil davon in einem Vortragsprogramm vorgestellt.
Verfügbar waren P 8000-Systeme mit WEGA, EC 1834 mit MUTOS 1834, A 7150 mit MUTOS 1700, VIDEOTON VT 32 mit DMOS, PC/AT-Kompatible mit XENIX und ESER mit PSU und VMX. Die Kopplung der Systeme über uucp-Komponenten wurde mit P 8000, EC 1834 und A 7100 demonstriert. Innerhalb des Vortragsprogramms informierten die Entwickler der Betriebssysteme über den Stand der Arbeiten und über die beabsichtigten Weiterentwicklungen:
– Robotron-Elektronik zu MUTOS 1700, MUTOS 1834, MUTOS 1800
– Technische Universität Karl-Marx-Stadt zu MUTOS 1835 und VMX
– ZFT des Kombinats EAW zu WEGA-Erweiterungssoftware.
Besonders wichtig war für den großen Kreis der Anwender, daß neben den Informationen zu Software und Hardware auch Probleme des Vertriebs und der Kundenbetreuung im direkten Kontakt mit den Herstellern diskutiert werden konnten.
Aus dem umfangreichen Angebot an Anwendersoftware sind hier folgende Schwerpunkte zu nennen:
– Datenbanksysteme (DABA32 für MUTOS 1800, ING_DB für MUTOS 1834 und andere Systeme, WEGA-DATA und SPHINX für WEGA)
– Compiler (CHILL für WEGA, PASCAL/C-Umsetzer für verschiedene Systeme, PROLOG für WEGA)
– Bildschirmbedienung (Masken- und Menügeneratoren unterschiedlicher Anwendungsorientierung für verschiedene Systeme)
– Grafiksoftware (2D- und 3D-Grafik für WEGA, XENIX und DMOS, Präsentationsgrafik).
Weiterhin wurden zahlreiche Dienstprogramme angeboten, u. a. für die Informations- und Dokumentationsverwaltung, Editierung und Textverarbeitung. Vorgestellt wurden auch erste Anwendungsprojekte. Sämtliche Softwareangebote, aber auch die Suchaufträge der Mitglieder, sind im Auskunftssystem der UNIX-EAG erfaßt worden und standen zur Recherche am Bildschirm zur Verfügung. Eine aktualisierte Version des Auskunftssystems wurde allen Mitgliedern übergeben.
Die Softwarebörse besuchten über 200 Teilnehmer aus 170 Institutionen der DDR. Der VEB Leitzentrum für Anwendungsforschung wird auf Grund der großen Resonanz auch 1989 eine solche Veranstaltung durchführen und versuchen, durch geeignete Organisation die Teilnahme aller Interessenten zu gewährleisten. Zugleich bittet der Veranstalter auf diesem Wege um Informationen zu Softwareangeboten.

H. Grützbach, VEB LfA

Mikroelektronik '88

Die 6. Fachtagung *des Fachverbandes Elektrotechnik der KDT*, die als gemeinsame Veranstaltung „Mikroelektronik '88" der Ingenieurorganisationen sozialistischer Länder vom 7. bis 9. Dezember 1988 in Dresden-Niedersedlitz stattfand, informierte mit über 100 Vorträgen in 7 Arbeitssektionen über den Stand und die Tendenzen der Entwicklung und Anwendung der Mikroelektronik.
Etwa 50 Prozent der Vorträge machte mit Ergebnissen bekannt, die durch die Tätigkeit der Ingenieurorganisationen in der UdSSR, VRB, VRP, ČSSR und UVR erzielt worden sind.
Neben den Vorträgen sind die Beratungs- und Informationsmöglichkeiten hervorzuheben, die es den Teilnehmern gestatteten, sich
– mit dem aktuellen Sortiment moderner elektronischer Bauelemente aus der DDR vertraut zu machen
– über 20 nachnutzbare mikroelektronische Lösungen in Form einer Posterdiskussion zu informieren und
– die im staatlichen Informationsdienst „Mikroelektronik" der DDR erfaßten Informationsquellen kennenzulernen.
In der Arbeitssektion *Rechnerbaugruppen und Kleincomputer* konnte über wesentliche Fortschritte berichtet werden, die seit der 5. Fachtagung im Jahr 1986 erreicht worden sind. Neben den Ausführungen zum „Programmiermodul EPROM 1" und dem „NANOS-Gerätesystem" standen vor allem die Vorträge zur Weiterentwicklung der Kleinrechentechnik im Mittelpunkt des Interesses. Dabei kann die Vorstellung des Bildungscomputers A 5105 als besonders gelungen angesehen werden.
Über die Nutzung von Kleinrechnern wurde in mehreren Vorträgen berichtet. So informierte ein Fachkollege aus Kaunas (UdSSR) über die praktizierte Erarbeitung von Lehrprogrammen für die Ingenieurausbildung mittels Rechner. Über den Einsatz des KC 85 in der CAD-Ausbildung wurde aus Mühlhausen (DDR) berichtet. Dabei wurde darauf hingewiesen, daß sich der Einsatz von Rechnern in der Ausbildung in den nächsten Jahren verändern wird, da mit immer größerer Leistungsfähigkeit der Rechner und der komplizierter werdenden Hardware und Peripherie, verbunden mit aufwendigen Programmen, die Eigenprogrammierung eines Problems immer schwieriger wird. Damit ist die Verwendung in der allgemeinen Ausbildung kaum noch möglich. Im Vordergrund steht deshalb künftig, die Computertechnik den Auszubildenden als Werkzeug zu erläutern und nahe zu bringen.
Etwa in diese Richtung geht auch das Anliegen, das mit dem Bildcomputer A 5105 verfolgt wird. Mit diesem Gerät steht ein kostengünstiges und flexibles System zur Verfügung, das zur Informatikgrundausbildung an Schulen und als modernes Werkzeug zur Gestaltung des Unterrichtes genutzt werden kann.
Mit der Vorstellung der Industrie-Computer-Gerätefamilie ICA 700 wurde über ein anwendungsorientiertes und leistungsmäßig gestaffeltes System informiert, mit dem leistungsfähige Personalcomputer, gekoppelt mit einem Echtzeitrechner, für viele industrielle Anwendungsfelder und Automatisierungsvorhaben genutzt werden können. Innerhalb der Gerätereihen ICA 710 und ICA 720 sind dazu unterschiedliche Einsatzvarianten verfügbar (modulares Mikrorechnersystem auf Baugruppen- und Kassettenebene, Schrankausführung mit echtzeitfähiger Prozeßkoppeleinheit, begrenzt konfiguierbare Auftisch-(Pult-)Variante), die als integraler Bestandteil von Geräte- und Anlagensystemen für Problemlösungen, als Leitstationen oder auch als Zellenrechner eingesetzt werden können.
Die Vorträge und Diskussionen in den weiteren Arbeitssektionen der Tagung
– zu monolithisch- und hybridintegrierten Schaltkreisen
– zum Layout-Entwurf von VLSI-Bauelementen
– zu Fragen der Technologie, Prüfung und Zuverlässigkeit elektronischer Bauelemente sowie
– zu Rationalisierungs- und Automatisierungslösungen, ergänzt durch
– eine Posterdiskussion zu multivalent nutzbaren Lösungen,
ließen recht eindrucksvoll die seit der 5. Fachtagung „Applikation Mikroelektronik" im Jahre 1986 erreichten Ergebnisse erkennen, machten aber auch die noch weiter zu lösenden Aufgaben deutlich, um die weitere Entwicklung und Anwendung der Mikroelektronik zu beschleunigen und umfassend durchzusetzen.
Ergänzend zu den Hardware-Informationen wurde während der Tagung eine „Software-Börse" durchgeführt. Der dazu vom VEB Datenverarbeitungszentrum Dresden vorgestellte Speicher enthielt 1500 Software-Anwenderlösungen, die mittels eines AC 7150 und Drucker abrufbar waren. Die Interessenten hatten auf diese Weise Gelegenheit, sich vorrangig über konstruktive CAD/CAM-Lösungen und integrierte Systeme für 16-bit-Rechner zu informieren.

Dr. R. Schneider

Kolloquium zu datenbankgestützter CAD-Software

Am 10. Januar 1989 fand in Leipzig ein Kolloquium „Entwurf und Applikation datenbankgestützter CAD-Software" als Gemeinschaftsveranstaltung des BV der KDT Leipzig und des CAD/CAM-Zentrums im Kombinat Polygraph statt. Unter großem Interesse der etwa 250 Teilnehmer wurde eine Übersicht über den Stand und die Entwicklungstendenzen auf dem Gebiet der CAD-Technologie in der DDR gegeben.
Doz. Dr. Herden, Leiter des CAD/CAM-Zentrums im Kombinat Polygraph Leipzig, wies in seinem Plenarvortrag auf die unbedingte Notwendigkeit einer abgestimmten Eigenentwicklung leistungsfähiger und modifizierbarer CAD-Software hin, um auf dieser Bais eine möglichst durchgängige Rechnerunterstützung vom Erzeugnisentwurf bis zur NC-gesteuerten Fertigung auf breiter Basis zu gewährleisten.
Ohne CAD kein CIM – unter dieser Prämisse wurden Zielstellung und Inhalt datenbankgestützter, über den Anwendungsbereich der Zeichnungserstellung hinausgehender CAD-Softwaresysteme diskutiert. Als Beispiel für eine derartige Systementwicklung wurde das CAD-Paket PolyCAD vorgestellt, das im CAD/CAM-Zentrum Polygraph in Zusammenarbeit mit Robotron und der TU Karl-Marx-Stadt entwickelt wird.
PolyCAD ist auf Mikrorechnern des Typs A 7150 G, EC 1834, und XT/AT-kompatiblen Rechnern unter MS-DOS lauffähig. Dieses CAD-Paket orientiert sich hinsichtlich wesentlicher Grundfunktionen am Leistungsniveau international bekannter Systeme und ermöglicht eine weitgehende Anpassung an spezifische Aufgabenstellungen im Bereich der Konstruktion, Projektierung und Technologie auf der Grundlage einer integrierten Datenbank.
Mit einem Beitrag der TU Karl-Marx-Stadt zum Problemkreis der technischen Modellierung wurde auf eine wesentliche Voraussetzung für eine effektive, ESKD-gerechte Zeichnungserstellung und NC-Programmierung hingewiesen. Zu diesem Problemkreis wurde eine NC-Schnittstelle im CAD-System PolyCAD vorgestellt. Einen besonderen Akzent setzte *Dr. Loeschke* vom Kombinat Wälzlager und Normteile mit seinem Vortrag zur notwendigen Integration von Expertensystemen in den CAD-Bereich. Am konkreten Beispiel eines Expertensystems zur Wälzlagerauswahl wurden die weitreichenden Möglichkeiten deutlich, die Arbeit des Konstrukteurs durch effiziente Lösungswege und gespeichertes Expertenwissen wirkungsvoll zu unterstützen.

W. Tischer

Wirksamer sprechen – hören – sehen

(Technik, Methodik und Praxis der Informationsaufnahme und -abgabe) von Wolfgang Zielke, 77 Seiten, 23 Literaturstellen. Grafenau/Württ.: Expert Verlag 1984.

„Seltsamerweise ist bei allem technischen Fortschritt ... das Feld der geistigen Arbeit lange Zeit unbeachtet geblieben" sagen Verlag und Verfasser im Vorwort und wollen mit dem Titel dazu beitragen, „beträchtliche Mühe und Zeit (bei Arbeitstechniken und Vorgehensweisen besagter Aufnahme und Abgabe) ein(zu)sparen". Die Strukturierung (einschließlich die Typographie) des Stoffes (Kapiteltitel: Denkarbeit und Information"; „Sprechend überzeugend wirken"; „Gutes Hören bringt viel ein" „Rechtes Sehen – rechtes Lesen") ist hervorragend gelungen, der Stil angenehm lesbar, die nützliche Redundanz angemessen.
Man spürt in jeder Zeile den Praktiker. Es ist kein Buch für den Profi; der „Einsteiger" findet hier eine überschaubare Grundlage für das informationsverarbeitende System „Mensch".

Horst Weichardt

Technologien im Umbruch

von W. Sydow (Hrsg.), Verlag Die Wirtschaft, Berlin, 1988, 352 S., 65 Abbildungen, 32 Fotos, 24 Tabellen, DDR 24,80 M

In einem interessanten Vorhaben – gemäß internationalen Tendenzen – hat der Herausgeber des Titels „Technologien im Umbruch" eine Reihe prominenter und sehr profilierter Experten zusammengeführt, um eine bemerkenswerte Repräsentation aktueller Erkenntnisse und Auffassungen zum genannten Gebiet zu vermitteln. Mit den geschickt ausgewählten 19 Beiträgen zu zentralen Begriffen unserer Zeit erhält der Leser eine ausgezeichnete Einführung und Übersicht: Schlüsseltechnologien – mikroelektronische Schaltkreisintegration – Technologie und Umwelt – Qualifikation – Arbeitskräfte – Menschen und Natur – fünfte Rechnergeneration – Informationstechnologien – Roboter – Energien – Biotechnologie – friedlicher Weltraum u. a. Notwendigerweise enthält das Buch viel Bekanntes. Insofern bildet es in manchen Fragen bereits einen Status. Aber nur selten wird Zukunft aus weit zurückliegender Vergangenheit gerechtfertigt. Manche Dinge sind für die meisten Leser sehr neu, so das Kapitel zur 5. Rechnergeneration von Moto-Oka, die Gedanken von Moissejew und Frolow zur Theorie der Noosphäre des genialen sowjetischen Gelehrten Wernadski oder die Theorie der „langen Wellen" der Wirtschaftsentwicklung infolge technologischer Umbrüche von Schumpeter. Zahlreiche Begriffe und Zusammenhänge erscheinen originell gefaßt und folgerichtig begründet. So zum Beispiel der Zusammenhang von „Werkzeug" und „Denkzeug" bezüglich der industriellen bzw. wissenschaftlich-technischen Revolution, die Rolle der Vernetzung, die Einheit von Informations- und Prozeßtechnik, die flexible Automatisierung, die Einheit von Erzeugnis und seiner Fertigungstechnologie, auch Schlüsseltechnologien und sozialer Fortschritt, „Schatz von objektiven Erkenntnissen", abproduktfreie Technologien und Umwelt, Schlüsseltechnologien und Qualifikationsstruktur (immer noch sind in der DDR 40% aller Beschäftigten in der unmittelbaren Produktion nötig), Innovationen und Wirtschaftswachstum, Elektronenstrahl-Veredelungstechnologien, hohe Erwartungen in die Biotechnologie.

Nahezu alle Beiträge sind in einer offenen und sehr gut lesbaren Sprache geschrieben. Bilder, Tabellen, Fotos und zahlreiche Zahlenangaben ergänzen in ansprechender Weise den Text. Das Buch ist unbedingt eine lohnende Lektüre und für manchen ein nützliches Lehrbuch.

Prof. Dr. Dr. M. Roth

MS-DOS-Handbuch

von R. A. King, 5., überarbeitete Auflage, Sybex-Verlag, Düsseldorf 1987 ISBN 3–88745–644–0

Das Buch ist sowohl als Handbuch für den lernenden sowie versierten MSDOS-Anwender, aber auch als Nachschlagewerk für den Systemspezialisten geeignet. Der lockere,



zum Teil belletristisch anmutende Stil des Autors erleichtert Neulingen den Zugang zum Problemkreis „MSDOS". Das Buch ist didaktisch übersichtlich aufgebaut, es enthält ausführliche Hinweise für eine zweckentsprechende, effiziente Handhabung durch den Leser. Die Breite der behandelten Problematik ist groß, aber trotzdem zumeist tiefschürfend genug, um grundsätzliche Sachverhalte korrekt darzustellen. Während in den Kapiteln 1–9 systeminterne Zusammenhänge erläutert werden, sind – so der Autor – die Kapitel 10 bis 17 vor allem für den Anwender bestimmt.
Die Anwendung und Wirkung der einzelnen BIOS-Funktionen und MSDOS-Funktionsaufrufe ist ausführlich beschrieben und nochmals in einer Übersicht zusammenfassend dargestellt.
Teilweise wünschte man sich mehr Angaben in tabellarischer Form (z. B. bei Erläuterung bestimmter Steuerblöcke) und mehr grafische Darstellungen zugunsten einer Reduzierung erklärender Texte (z.B. beim Problem file allocation table/cluster selection), Als störend wurden kleine Unkorrektheiten bei Angaben von Verweisen auf numerierte Kapitel und nicht vorhandene Tabellen (z. B. program segment prefix) empfunden.
Hervorgehoben werden muß dagegen die sehr gut mit Beispielen unterlegte, ausführliche Beschreibung der Stapelverarbeitungs-Technik und die relativ geschlossene Darstellung der Fehlerbehandlung ab MSDOS-Version 3. Auch die Herausarbeitung der Unterschiede zwischen den einzelnen Betriebssystem-Versionen innerhalb MSDOS bzw. zwischen MSDOS und PCDOS verdient lobende Erwähnung.
Das Buch enthält an zahlreichen Stellen wichtige Hinweise aus der Praxis des Autors, die im angeführten Zusammenhang dem ungeübten Anwender oftmals zeitaufwendige „Knobeleien" ersparen. Das Werk geht damit als Handbuch inhaltlich weit über die von vielen Computer-Lieferanten bereitgestellte Literatur hinaus. Es ist daher für alle Personal-Computer-Einsteiger, aber auch für Fortgeschrittene als zur Ergänzung der „Maschinen-Literatur" geeignetes Hilfsmittel anzusehen.

D. Lenz/M. Roth

Encyclopedia of microcomputers

Hersg. Allan Kent; James G. Williams, New York; Basel: Dekker
Vol.1. 1987. – VIII, 434 S.
ISBN 0–8247–27002
Vol. 2. 1988. – VII, 452 S.
ISBN 0–8247–2701–0

Herausgeber und Verlag sind international wohl renommiert. Mit den bisher 42 Bänden der „Encyclopedia of Library and Information Science" und den 16 Bänden der „Encyclopedia of Computer Science and Technology" liegen auch große lexikographische Erfahrungen vor.
Die Enzyklopädie ist auf 10 Bände veranschlagt, in denen ca. 500 Artikel auf etwa 5000 Seiten veröffentlicht werden sollen. Herausgeber und Verlag orientieren sich auf die breiten Leserschichten von Hardware-Spezialisten, Programmierern, Mathematikern. Ihnen sollen Nachschlagemöglichkeiten sowohl zu Mikrocomputertechnologien als auch der Nutzung solcher Technologien gegeben werden. Gedacht wurde auch an Interessenten aus anderen Disziplinen, die beruflich stark in die Arbeit mit Mikrocomputern integriert sind.
Vielleicht wird es in Zukunft durch die Computerentwicklung selbst zu solch elektronischen Enzyklopädien kommen, die vom einzelnen Leser bei der Nutzung seinem konkreten Wissensstand angepaßt werden können und flexibel mit unterschiedlichen Angaben zum einzelnen Stichwort Auskunft geben.
Von den geplanten 10 Bänden liegen die ersten beiden vor, Band 1 umfaßt 26 Artikel, von Access Methods zu Assembly Language und Assemblers.
Band 2 enthält 25 Artikel, von Authoring Systems for interactive Video zu Compiler Design.
Die einzelnen Beiträge sind nach Möglichkeit folgendermaßen aufgebaut: Geschichtliche, gegenwärtige und zukünftige Aspekte, mit Definition oder wenigstens definitorischer Beschreibung. Angeschlossen sind umfangreiche (englischsprachige) Literaturhinweise.
Bei einem Vergleich der im ersten Band ausgewählten Stichworte finden wir allein sieben Firmenbeschreibungen aus der Mikrocomputer- und softwarebranche, ähnliche Proportionen treten auch im zweiten Band auf. Für den Nutzer außerhalb der USA bietet sich damit ein bewerteter Überblick über bedeutende Firmen.
Da jeder Rezensent aus dem Umfeld seiner beruflichen Erfahrungen bewertet, hätten Rezensenten aus den ebenfalls angesprochenen Nutzerkategorien sicher andere Schwerpunkte hervorgehoben, Wünsche geäußert. Abschließend sei aber die vorzügliche äußere und typographische Gestaltung, die Kunstdruckpapierqualität hervorgehoben.

Dr. Jürgen Freytag

in FOKUSO 3/88:

Deak Jahn:
Unverzichtbare Peripherie – die Drucker
(Haupttypen der Drucker, Sortiment, Funktionsweise)
T. Kolossa:
Laserdrucker
(Funktionsprinzip, Typen, editierende Sprachen, Zeichensatz)
S. Szabo:
Neuro-Computer – Traum oder Wirklichkeit?
(Math. Modell des Gehirns, gehirnähnliche Computer, Anwendung)
L. Kovacs:
Neue Ergebnisse der Sprachübersetzung mittels Computers
(Entwicklung von Übersetzungssystemen, Zwischensprache oder Übergang als Methode, experimentelle Systeme, Terminologie)

Internationale populärwissenschaftliche Zeitschrift in Esperanto, herausgegeben vom Ungarischen Esperanto-Verband, 4 Hefte à 64 S. im Jahr, Preis pro Jahrgang 24,– M, beziehbar in der DDR über M. Behr, Koburger Str. 83, Markkleeberg, 7113

B.


ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 5 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
Mikroprozessorsystem
U 80600

Kristallzüchtungsanlage für 150-mm-Wafer

Im Treptower Betriebsteil des VEB Steremat „Hermann Schlimme" Berlin – ein Betrieb des Kombinats Elektro-Apparate-Werke – wurden am 20. Februar dieses Jahres zwei Muster der neuen Kristallzüchtungsanlage CZA 2000 vorgestellt. Die CZA 2000 wird es den Schaltkreisherstellern in der DDR ermöglichen, Siliziumscheiben bis zu einem Durchmesser von 150 mm zu produzieren. Aus einem Bericht der Berliner Zeitung geht hervor, daß die neue Kristallzüchtungsanlage bis zum 7. Oktober, dem 40. Jahrestag der DDR, in die Produktion überführt werden soll. In Anwesenheit des 1. Sekretärs der Bezirksleitung der SED Berlin, Günter Schabowski, berieten Arbeiter, Leiter, Forscher und junge Kandidaten zur Kommunalwahl darüber, wie die anspruchsvollen Aufgaben gelöst werden können. Unter anderem wird für die Produktion der CZA 2000 im Treptower Betriebsteil von Steremat eine neue Produktionshalle errichtet. *MP*

Wie klappt der Computerservice?

Dr. Manfred Schröder, Stellvertreter des Generaldirektors des VEB Kombinat Robotron für Binnenmarktvertrieb und Kundendienst, erläuterte dazu in Presseinformationen 152/1988:

Erste Erfahrungen zeigen, daß sich die Arbeitsteilung zwischen dem Kombinat Robotron und den Anwendern bei Instandhaltung von Computern schon gut bewährt. Gegenwärtig betreuen 600 Anwender und -gemeinschaften die von ihnen genutzte Technik selbst, und zwar auf der Grundlage von Vereinbarungen mit dem Kombinat Robotron. Das geschieht zweigorientiert, im Kombinatsmaßstab oder konzentriert im Territorium. Auf diese Weise wurden eine deutliche Leistungssteigerung im Kundendienst und eine spürbar größere Verfügbarkeit der Computer erreicht. Im Durchschnitt gelingt es derzeit in der Volkswirtschaft, 90 Prozent der Reparaturen innerhalb von 6 Werktagen zu erledigen, weit über die Hälfte davon sogar innerhalb von 2 Werktagen.

Die vier Kundendienstbetriebe des Kombinates Robotron in Berlin, Leipzig, Erfurt und Karl-Marx-Stadt schaffen in allen Teilen der DDR maßgebliche Voraussetzungen für die arbeitsteilige Computerinstandhaltung. Dazu gehören unter anderem die Ausbildung von Technikern, die Übergabe technologischer Unterlagen, die Versorgung mit spezifischer Meß- und Prüftechnik. Das Kombinat stellte allein 1988 etwa 15000 Lehrgangsplätze bereit, um Werktätige für den Computerservice zu qualifizieren. Gegenwärtig erweitern die Kundendienstbetriebe ihre Kapazitäten für die Reparatur von Baugruppen, arbeiten an verbesserten Diagnosemöglichkeiten und bereiten alles vor, um die neue Generation von Personal- und Arbeitsplatzcomputern mit 16 bzw. 32 Bit Verarbeitungsbreite in die arbeitsteilige Instandhaltung einzubeziehen.

Bei allem richtet Robotron großes Augenmerk auf die Produktion von Datenverarbeitungsanlagen und Computern für die Volkswirtschaft in guter Qualität. Zusammen mit einem leistungsfähigen technischen Kundendienst wirkt sich das günstig auf die Effektivität aus.

Der eingeschlagene Weg der Arbeitsteilung erweist sich als effektiv, auf dieser Basis ist es möglich, die volkswirtschaftlichen Anforderungen an den Kundendienst für CAD/CAM- und Rechentechnik auch langfristig zu erfüllen



Bulgarische Computertechnik in Berlin

Gemeinsam mit der bulgarischen Außenhandelsorganisation Isotimpex veranstaltete die Wirtschaftsvereinigung Mikroprozessorsysteme – Prawez im Februar in Berlin eine Ausstellung, die einen guten Querschnitt ihrer Erzeugnisse bot (siehe Bild oben). Der Schwerpunkt lag dabei nicht in der Präsentation auffälliger Spitzenprodukte – in unseren Berichten von der LFM '88 und der Technischen Messe Plowdiw '88 hatten wir zum Beispiel über neue 16- und 32-Bit-PCs aus Prawez berichtet –, sondern im Darstellen der Möglichkeiten eines umfassenden Einsatzes der Computertechnik in der Volkswirtschaft. Das Angebot reichte demzufolge von – durchweg sofort lieferbaren – Lösungen zum rechnergestützten Unterricht über spezifische Module zur Laborautomatisierung oder industriellen Anwendung bis zu Software der Künstlichen Intelligenz, geeignet zur Anwendung auf 16-Bit-PCs. Den letztgenannten Bereich repräsentierte beispielsweise eine auf dem EC 1839 (XT-kompatibel) basierende KI-Arbeitsstation (Bild unten). Die Software umfaßte eine vom Institut für Mikroprozessortechnik entwickelte Implementation von Prolog, IMS-Prolog, sowie verschiedene Expertensysteme: SUPREME, ein



Fotos: Weiß

System zur Entscheidungsfindung in gering strukturierten Bereichen und unter Risikobedingungen, Cortech, ein Expertensystem zum Steuern von technologischen Prozessen, und Agrotech, ein Expertensystem für Agrartechnologien.

Einen größeren Umfang nahmen die Möglichkeiten zur Vernetzung der PCs ein. In unserem Bericht von der LFM '88 hatten wir bereits auf das lokale Netz Microstar hingewiesen; weitere gezeigte Netze waren Microring, Microlim, ISNET und ESPANET.

Letztlich sei noch auf das Angebot von CAD-Software hingewiesen, die bereits am XT-kompatiblen PC den Entwurf einfacher Leiterkarten ermöglicht. *MP-We*

Computer an USA-Schulen

	Schüler pro Computer	Computer pro Schule	Anzahl der Schulen	Rang unter den Staaten
Durchschnitt	31,87	15,50	80876	–
New York	23,76	28,34	3884	5
Californien	38,72	14,89	7723	44
Alaska	18,09	15,58	426	2
Wyoming	16,54	16,72	355	1
Lousiana	52,08	10,20	1871	50
Mississippi	56,26	9,31	917	51

Die Wachstumsrate der Computeranzahlen beträgt im Mittel 19% je Jahr. (nach Interface, Schweiz 1988, H. 2)

Im Schuljahr 1987/88 gab es in den USA 40 Millionen Schüler in etwa 80000 Schulen (Elementar-, Mittel- und Oberstufe), denen 1,24 Mill. Computer zur Verfügung standen. In entwickelten und stark industrialisierten Ländern kann man mit 2500 bis 3000 Einwohnern pro Schule rechnen. Das Verhältnis Einwohner : Schüler ist sehr unterschiedlich. Während in der DDR 8 : 1 gilt (etwa 2 Mill. Schüler oder 200000 pro Jahrgang), ist dies in den USA 5 : 1. Daraus folgt, daß es in den USA Bundesstaaten mit zurückgebliebener Entwicklung gibt, und daß die Schulen und Klassen in den USA zumindest in diesen Gebieten höhere Schülerbesetzungen haben als das in der DDR der Fall ist. Die Auswahl der Informationen in der Tafel bestätigt das. Das dünnbesiedelte Alaska hat nur 18 Schüler pro Computer, weil dort wegen der extremen Winter Schüler die Schulen nicht erreichen können und man zeitweilig zum dezentralisierten Unterricht mit Computerunterstützung (Disketten) übergeht. Die angegebenen Zahlen „Schüler pro Computer" beziehen sich auf die Gesamtzahl der Schüler, nicht etwa nur auf die Jahrgänge mit Unterricht über Computer, weil sehr weitgehend – und auch schon in der Elementarstufe – Unterricht und selbständige Arbeit mit Computerunterstützung realisiert wird. Der Anteil der 8-Bit-Computer wird immer kleiner.

Prof. Dr. I. O. Kerner

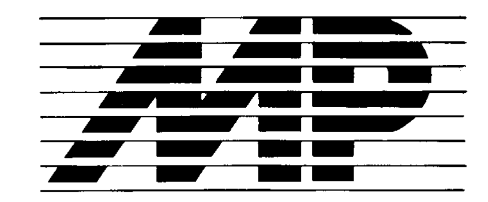
2. Tagung des RGW-Komitees für Elektronisierung

Das Komitee des RGW für die Zusammenarbeit auf dem Gebiet der Elektronisierung tagte Anfang November 1988 in Warna. Ausgehend von den Festlegungen des Komplexprogramms des wissenschaftlich-technischen Fortschritts bis zum Jahr 2000 stand im Mittelpunkt der Beratung die Intensivierung der Zusammenarbeit auf dem Gebiet der Entwicklung und Produktion mikroelektronischer Bauelemente, einschließlich der zur Produktion erforderlichen technologischen Spezialausrüstungen sowie Spezial- und Hilfsmaterialien. Die Delegation der DDR wurde vom Minister für Elektrotechnik und Elektronik, Felix Meier, geleitet. *ADN*

Direkte Computerverbindung zwischen UdSSR und USA

Eine Computer-Fernverbindung zwischen der UdSSR und den USA ist kürzlich erstmals von der USA-Gesellschaft San-Francisco-Moscow-Teleport und dem Forschungsinstitut für angewandte automatisierte Systeme – der Dachorganisation der UdSSR für die automatische Verbindung mit Computernetzen in anderen Ländern – hergestellt worden. Bei der Probe-Nachrichtenübertragung wurde der internationale Nachrichtensatellit „Intelsat" als Zwischensender eingesetzt. Damit ist die UdSSR in der Lage, die Computernetze aller sozialistischen Länder sowie Österreichs und Finnlands zu nutzen. *ADN*

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232-2892



Mikroprozessortechnik, Heft 5 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelfoto Eberhard Mai

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

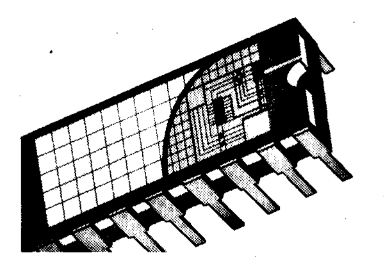
Redaktionsschluß: 13. März 1989

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

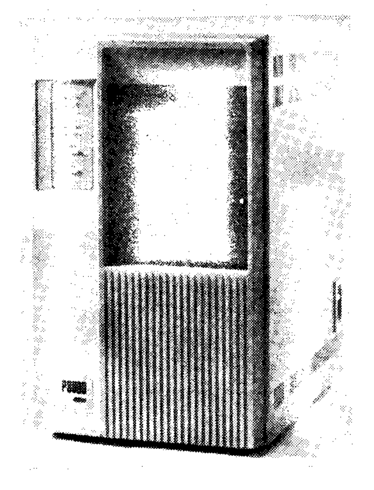
Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



13. Mikroelektronik-Bauelementesymposium

Die ersten Schaltkreise des schnellen 16-Bit-Mikroprozessorsystems vom VEB Kombinat Mikroelektronik Erfurt wurden auf der Leipziger Frühjahrsmesse vom 12. bis 18. März 1989 erstmals gezeigt. Das System U 80600, das einmal elf Schaltkreistypen umfassen soll, wurde zum 13. Mikroelektronik-Bauelementesymposium vom 8. bis 10. Mai 1989 in Frankfurt (Oder) durch Vorträge ausführlich vorgestellt. Wir haben für Sie Kurzfassungen der Vorträge zum System U 80600 sowie zu den Schaltkreisen CPU (U 80601), Buscontroller (U 80606), EDC (U 80608), DRAM-Controller (U 80610), SCC (U 82530) und CIO (U 82536) vorbereitet.
Auf der vierten Umschlagseite finden Sie Abbildungen und Kurzcharakteristiken der vier Schaltkreise des Systems U 80600, die auf der LFM '89 zum erstenmal zu sehen waren.



Als erste Anwendung des Systems U 80600 stellen wir auf der **Seite 160** den Rechner P8000 compact vom VEB Kombinat Elektro-Apparate-Werke Berlin vor, der auf der LFM '89 ausgestellt war. Er beinhaltet neben den CPUs U 880 und U 8000 die vier Schaltkreise U 80601, U 80606, U 80608 und U 80610. Dadurch wurde gegenüber dem Vorgänger, dem P8000, die Möglichkeit geschaffen, außer den Betriebssystemen UDOS und WEGA auch das Betriebssystem WDOS, das teilkompatibel zu MS-DOS ist, nutzen zu können.

Vorschau

In MP 6/1989 finden Sie unter anderem Beiträge zu folgenden Themen:
- Filesharing
- Künstliche Intelligenz
- U 8272 im Nicht-DMA-Modus
- externe Assemblerprogramme in Turbo-Pascal.

Inhalt

Das schnelle 16-Bit-Mikroprozessorsystem U 80600

Dr. Dietrich Mandler,
Dr. Hendrik Berndt
VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum

Auf der Grundlage der im VEB Mikroelektronik „Karl Marx" Erfurt realisierbaren VLSI-Technologien wird ein neues leistungsstarkes 16-Bit-Mikroprozessorsystem entwikkelt.
An den Systemarbeiten sind Entwurfszentren von Anwenderkombinaten und weiteren Praxispartnern, unter anderem das Kombinat Robotron Dresden, das Kombinat Carl Zeiss JENA, das Zentralinstitut für Kybernetik und Informationsprozesse Berlin, das Kombinat Elektro-Apparate-Werke „Friedrich Ebert" Berlin, das Kombinat Automatisierungsanlagenbau Berlin und das Halbleiterwerk Frankfurt (Oder) beteiligt und haben wesentliche Beiträge für die Schaltkreisentwicklung geleistet. Zielstellungen bei der Entwicklung des vorzustellenden schnellen 16-Bit-Mikroprozessorsystems waren die Implementierung moderner Architekturmerkmale und die Erreichung einer Abwärtskompatibilität zum 16-Bit-Mikroprozessorsystem K 1810 der UdSSR.

Vorbemerkung

Das System U 80600 stellt in der Leistungsfähigkeit seines Prozessors eine neue Generation der im VEB Mikroelektronik „Karl Marx" Erfurt gefertigten 16-Bit-Mikroprozessoren dar. Ihn charakterisieren dabei folgende Eigenschaften:

– interne, virtuelle Speicherverwaltung
– internes, leistungsfähiges Speicherschutzkonzept
– hohe Verarbeitungsgeschwindigkeit durch zwei Vorverarbeitungsmechanismen (auf Byte- und Befehlsniveau)
– leistungsfähige Adreßberechnung mittels zusätzlicher Arithmetikhardware
– hoher Datendurchsatz (Busschnittstelle mit 8 MByte/s).

Darüber hinaus verfügt das U 80600-System über eine wesentlich erhöhte Leistungsfähigkeit der Peripheriekomponenten. Eine Vergrößerung des komplexen Funktionsumfanges des Speichersystems und eine Erhöhung von Zuverlässigkeit und Nutzenskomfort werden durch hochintegrierte Systemkomponenten erreicht. Diese Leistungsmerkmale kennzeichnen die Hauptanwendungsfälle des U 80600-Systems in Personal- und Industriecomputern, in der Prozeßautomatisierung, in Arbeitsplatzsystemen und in Kommunikationssystemen, also für Einsatzbedingungen, bei denen eine hohe Verarbeitungsgeschwindigkeit und die Bearbeitung mehrerer Aufgabenstellungen gleichzeitig gefordert werden.

Systemumfang

Das Mikroprozessorsystem U 80600 ist insgesamt aus 20 Komponenten konzipiert, deren wesentliche Bestandteile in Tafel 1 dargestellt werden.
Die Systemleistung des U 80600-Systems soll einführend durch die Bauelemente

Tafel 1 System U 80600

Typ	Anwendung
U 80601	Mikroprozessor ≙ 80286
U 82720	Grafik-Display-Controller
U 82530	serieller Kommunikationscontroller
U 80606	Buscontroller
U 80608	Fehlererkennungs- und -korrekturschaltkreis
U 80610	programmierbarer DRAM-Controller
DS 82284	Taktgenerator
U 82536	Zähler-/Zeitgeber- und paralleler Ein-/Ausgabeschaltkreis
U 8272	Floppy-Disk-Controller
U 82062	Harddisk-Controller

U 80601 Mikroprozessor
U 80606 Buscontroller
U 80608 Schaltkreis zur Fehlererkennung und -korrektur
U 80610 DRAM-Controller
beschrieben werden.
Das System ist erweiterbar um
– Busarbiter
– DMA-Controller
– Arithmetikprozessor
– LAN-Komponenten
und andere Multifunktionsperipherieschaltkreise.

Für den Aufbau unterschiedlichster Konfigurationen stehen dem Anwender für den Ausbau des Gesamtsystems Adreßbus- und Datenbustreiber mit einer Breite von 8 Bit aus dem Halbleiterwerk Frankfurt (Oder) und weitere Schaltkreise des 16-Bit-Mikroprozessorsystems K 1810 WMxx zur Verfügung.
Damit ist eine im Bild 1 vom Prinzip her dargestellte Systemkonfiguration realisierbar, die (ausgerüstet mit Floppy-Einheiten, Festplatten, Anschlußmöglichkeit an lokale Netze u. a.) den gestellten Anforderungen der Geräteindustrie der DDR entspricht.

Eigenschaften des Mikroprozessors

Architektur

Die hohe Prozessorleistung des U 80601 wird durch vier parallel arbeitende Einheiten erreicht. Die **Buseinheit** organisiert die Zugriffe auf Speicher und Peripherie und den Datentransfer mit Koprozessoren. Außerdem übernimmt sie die DMA-Steuerung (Direct Memory Access = direkter Speicherzugriff). In dieser Einheit wird ebenfalls eine Bytevorverarbeitung durchgeführt.
In der Befehlseinheit erfolgt die Zusammenfassung der vorverarbeitenden Bytes zu Befehlen. Es wird dabei ein eigener Vorverarbeitungsmechanismus angewendet, der

– das Behandeln von Präfixen
– das Verlängern von Immediate-Operanden
– das Ermitteln der Mikroprogrammadresse

durchführt. Die Befehlswarteschlange umfaßt drei Befehle. Die **Ausführungseinheit** dient zur eigentlichen Befehlsbearbeitung. Sie enthält deshalb die Allzweckregister, die Arithmetik-Logik-Einheit, weitere Logikeinheiten zur Bearbeitung komplexer Befehle und den Mikroprogrammspeicher.
Mittels der **Adreßeinheit** erfolgt parallel zur Befehlsabarbeitung die Adreßberechnung.

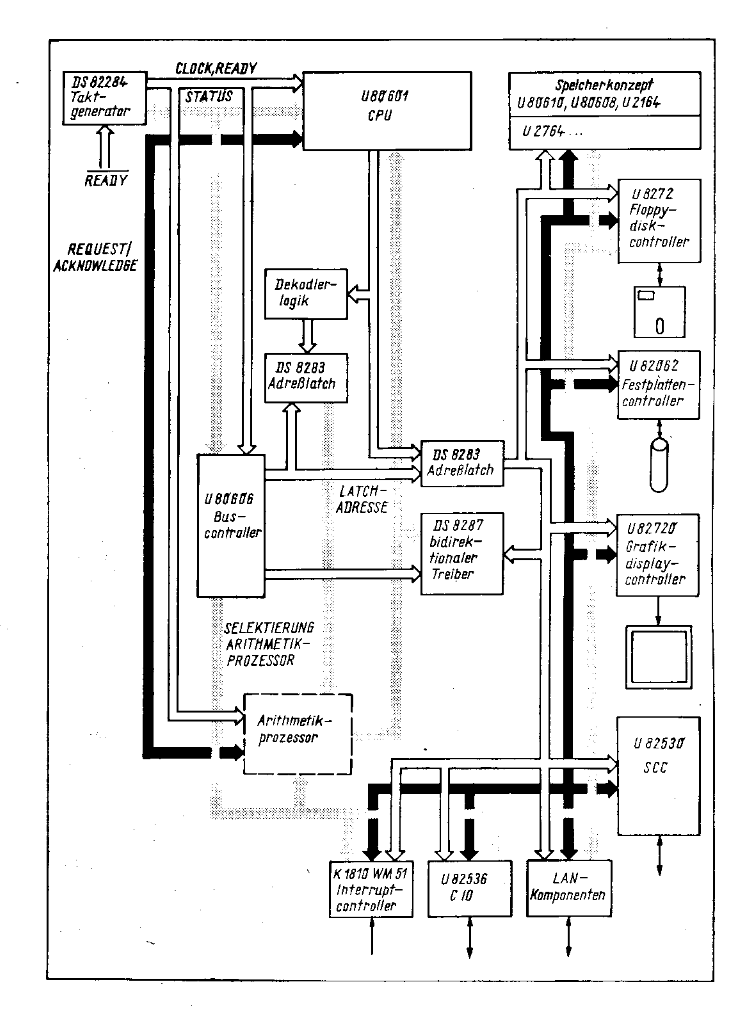


Bild 1
Systemkonfiguration des U 80600

Hierzu wird für jeden Buszyklus eine effektive Adresse innerhalb eines Segments gebildet. Nach Anwendung von Schutzmechanismen wird daraus eine logische Adresse abgeleitet. Diese dient wiederum zur Berechnung einer physischen Adresse nach dem Segmentierungsprinzip. In der Adreßeinheit ist eine Reihe von Cacheregistern vorhanden, die Informationen für die Segmentierung und die Schutzfunktionen beinhalten.

Architekturmerkmale

Virtuelle Adressierung

Die Aufteilung des Adreßraumes des U 80601 erfolgt in Segmente, deren Basisadresse beliebig im physischen Adreßraum verschiebbar ist. Es erfolgt eine Zuordnung der Segmente zu Code-, Daten- und Stackbereichen. Die Beschreibung der Segmente erfolgt wiederum durch Tabellen, sogenannte Segmentdeskriptortabellen. Der virtuelle Speicherraum wird mittels dieser Deskriptoren zu 1 GByte je Task festgelegt.

Zugriffsschutz

Es erfolgt eine Isolierung von Tasks, indem jeder Task eine eigene, lokale Tabelle von Segmentdeskriptoren zugeordnet werden kann. Über diese Deskriptoren werden unter anderem Zugriffsrechte spezifiziert, deren Verletzung zu Trapfunktionen führt.
Als zusätzliche Maßnahme zum Schutz von Betriebssystemfunktionen wurden beim U 80601 vier Privilegstufen definiert. Die Programmabarbeitung in den Privilegstufen und der Übergang auf andere Privilegstufen wird dabei durch eine Hardware im Prozessor unterstützt.

Integrierter Taskwechsel

Die Bedeutung von schnellen Taskwechselmechanismen ist vor allen Dingen in Echtzeitbetriebssystemen sowie bei Multitasking- und bei Multiuserfähigkeit von Betriebssystemen von großer Bedeutung. Im U 80601 ist deshalb ein hardwaregestützter Taskwechselmechanismus implementiert. Die Beschreibung von Tasks erfolgt dabei wiederum über Tabellen (44 Byte je Task). Ein Taskwechsel mit Registerrettung und Registerwechsel erfolgt innerhalb von 21 Mikrosekunden.

Betriebsarten

Der U 80601 besitzt zwei grundsätzliche Betriebsarten, den *Real Address Mode* und den *Protected Mode*. Nach dem Einschalten oder nach Systemreset befindet sich der Prozessor automatisch im Real Address Mode. Damit ist der U 80601 voll objektcodekompatibel zum K 1810 WM86. Es erfolgt eine reale Adressierung, der physische Speicherbereich umfaßt 1 Megabyte. Der Befehlssatz wurde um einige Befehle erweitert. Die CPU U 80601 arbeitet auf Grund des besseren Zykluszeitverhaltens wesentlich schneller.
Nach dem Aufbau der notwendigen Deskriptortabellen kann mittels eines speziellen Umschaltbefehls der Übergang in den Protected Mode erfolgen. In ihm werden die beschriebenen Archtitekturmerkmale, virtuelle Adressierung und Zugriffsschutz, aktiv. Der Arbeitsspeicher umfaßt 16 Megabyte, und der Befehlssatz wird außerdem um architekturabhängige Befehle erweitert. Für diese Betriebsart ist der U 80601 optimiert. Den U 80601 kann ein parallel zu seinen Adreß- und Datenleistungen liegender Koprozessor unterstützen.

Kurzcharakteristik ausgewählter Schaltkreise

Buscontroller U 80606

Der Einsatz des Buscontrollers U 80606 steigert die Leistungsfähigkeit des U 80601. Er dekodiert dessen Statussignale und stellt die wichtigsten Befehls- und Steuersignale für das Bussystem bereit.
Innerhalb des U 80600-Systems dient der U 80606 zum Erzeugen der Adreßlatch-Steuersignale der Datenübertragungssteuerung und zur Standard-Befehlsausgabe. Dabei wird für zwei Betriebsarten (MMS-16-Bus, lokaler Bus) das Zeitverhalten gesteuert. Durch zwei Buscontroller ist die Arbeit des U 80601 mit zwei Bussystemen durch Auftrennen des physischen Adreßraumes möglich.

Schaltkreis zur Fehlererkennung und -korrektur U 80608

Der Fehlererkennungs- und -korrekturschaltkreis U 80608 ist ein auf die Erhöhung der Zuverlässigkeit von Speichersystemen ausgerichteter Schaltkreis. Er dient zur schnellen Fehlererkennung und -korrektur in statischen und dynamischen Speichersystemen.
Der U 80608 ist in der Lage, 8 oder 16 Datenbits und bis zu 8 Prüfbits zu verarbeiten. Durch Kaskadierung von maximal 5 Schaltkreisen dieses Typs können Datenworte bis zu einer Breite von 80 Bit verarbeitet werden.

DRAM-Controller U 80610

Der U 80610 ist ein programmierbarer DRAM-Controller für dynamische Speicherkonzepte. Er unterstützt den Anschluß von 16-, 64- und 256-KBit-DRAMs und ist in der Lage, einen Adreßraum von bis zu 2 MByte zu verwalten. Der Controllerschaltkreis gestattet RAM-Zugriffe ohne Wartezyklen, gestattet eine RAM-Initialisierung und die Programmierung von 4 Refreshmodi. Durch seinen Einsatz reduziert sich der zur Ansteuerung dynamischer Speicher erforderliche Hardwareaufwand erheblich.
Der U 80610 gestattet mit seinen 2 Ports den Aufbau von Dual-Port-RAM-Strukturen bei synchroner oder asynchroner Kommunikation mit dem Prozessor. Darüber hinaus verfügt der U 80610 über ein ECC-Interface. In Verbindung mit dem Fehlererkennungs- und -korrekturschaltkreis U 80608 gelingt damit der einfache Aufbau großer Speichersysteme, die fehlerkorrigierende Codes erzeugen und korrigierbare RAM-Fehler während der Refreshphase beseitigen.
Gegenüber bisherigen Mikroprozessorsystemen (U 880, U 8000, K 1810 WMxx), die nur eine Fehlererkennung, aber keine Fehlerkorrektur erlauben, stellt dies eine wesentliche Verbesserung der Systemzuverlässigkeit dar.

Ausblick

Der VEB Mikroelektronik „Karl Marx" Erfurt erweitert und verbessert durch Einführung neuester VLSI-Technologie das Sortiment an verfügbaren und neu eingeordneten Bauelementen des U 80600-Systems. Er verwirklicht damit ein kontinuierliches Konzept der Weiterentwicklung der Bauelemente, das sich am Bedarf der Anwenderindustrie und an den wachsenden Anforderungen zukünftiger Systemlösungen orientiert.

⊠ KONTAKT ☎

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76, App. 57

TERMINE

2. Internationale Fachtagung „Lichttechnik"

WER? Fachverband Elektrotechnik in der Kammer der Technik, Wissenschaftliche Sektion Lichttechnik, Kombinat VEB NARVA
WANN? 1. Quartal 1990
WO? Berlin
WAS?
- neue Entwicklungen
- rationelle Energieanwendung
- Einsatz der Mikroelektronik und Rechentechnik bei der Erzeugisentwicklung von optischen Strahlungsquellen und Leuchten sowie in der Beleuchtungstechnik, insbesondere bei Projektierungsaufgaben (CAD/CAM)
- Einsatz der optischen Strahlungstechnik für Hochtechnologien (Mikroelektronik, Biotechnologie, Medizin u. a.)
- neueste Erkenntnisse der Grundlagenforschung.

WIE? Zu diesen Themen bitten wir um Vorschläge für Vorträge und Poster bis zum 31. Mai 1989. Teilnahmemeldungen sind schriftlich an folgende Anschrift zu richten: Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Straße 115/117, Berlin, 1086; Tel. 2 26 52 85

Hoppe

Fachtagung „Mikroelektronische Steuerungen mittleren und größeren Umfanges"

WER? Arbeitsausschuß Steuerungs- und Regelungstechnik Dresden
WANN? 23. Mai 1989
WO? Dresden, Neues Rathaus
WAS?
- Speicherprogrammierbare Steuerungen
- Modulares Automatisierungssystem S4000
- Einsatz des Systems S2000-S
- Verbundsystem aus MRS 700 und 16-Bit-Leitrechner für eine Transport- und Lagerautomatisierung
- Anwendung modularer Grundbaugruppen des Systems 7000
- Industriecomputer
- Anwendung des ICA 710.20
- Weiterentwicklung des Prozeßleitsystems audatec mit 16-Bit-Mikrorechner
- P8000 compact

WIE? Anfragen richten Sie bitte an: VEB Bergmann-Borsig, TVE 2.2, Wasastraße 50, Radebeul, 8122; Tel. 71 36 31 (Koll. Schwalbe)

Prof. Dr. Habiger

16-Bit-Mikroprozessor U 80601

Andreas Fritzsche
VEB Mikroelektronik „Karl Marx“
Erfurt, Forschungszentrum

Vorbemerkungen

Die CPU U 80601 ist ein moderner, leistungsfähiger Mikroprozessor mit der Möglichkeit des Einsatzes in Multiuser- und Multitasksystemen. Abhängig von seinem Einsatz ist die Leistungsfähigkeit des U 80601 im Vergleich zur CPU K 1810 WM86 bis zu sechsmal größer, wobei der Prozessor softwaremäßig abwärtskompatibel zur CPU K 1810 WM86 ist.

Der U 80601 ermöglicht zwei Betriebsarten, den direkten Adreß-Modus (Real Mode) und den geschützten, virtuellen Adreß-Modus (Virtual Protected Mode oder Protected Mode). Beide Betriebsarten sind mit Hilfe des K 1810 WM86-Befehlssatzes programmierbar.

Programme des Real Modes nutzen einen (direkt adressierbaren) Adreßraum von bis zu einem Megabyte, während der U 80601 im Protected Mode automatisch einen virtuellen Adreßraum von bis zu einem Gigabyte pro Task, bestehend aus direkt adressierbaren Blöcken von je 16 MByte, bereitstellt.

Darüber hinaus existiert im Protected Mode ein Speicherschutz, der es beispielsweise ermöglicht, das Betriebssystem von Nutzerprogrammen zu trennen und verschiedene Tasks sowohl programm- als auch datenmäßig unterschiedlich privilegiert zu bearbeiten.

Beide Betriebsarten benutzen den gleichen Grundbefehlssatz, gleiche Register und Adressierungsmodi.

Baugruppen des U 80601

Entsprechend dem Blockschaltbild des U 80601 (Bild 1) läßt sich der Schaltkreis in folgende Baugruppen unterteilen:

- Buseinheit (Bus Unit – BU)
- Befehlseinheit (Instruction Unit – IU)
- Ausführungseinheit (Execution Unit – EU)
- Adreßeinheit (Address Unit – AU).

Buseinheit

Die Buseinheit enthält Adreßlatches und Treiber, das Koprozessorinterface sowie die Datenrichtungs- und die Bussteuerung für Speicher- und I/O-Zugriffe des U 80601 und zur Koordination mit Koprozessoren sowie zur Busvergabe an masterfähige Prozessoren.

Die Einheit arbeitet nach dem sogenannten Prefetch-Mechanismus, der bei unmittelbar aufeinanderfolgenden Befehlen bereits einen Vorgriff auf den folgenden Befehl ausführt. Falls innerhalb einer Befehlsfolge eine Lücke auftritt, wird die Prefetch-Schlange auf Null zurückgesetzt, und der Prefetch-Mechanismus setzt mit Beginn einer weiteren Befehlsfolge erneut ein.

Die 6 Byte tiefe Prefetch-Schlange enthält Befehlswörter, die momentan von der Ausführungseinheit noch nicht benötigt werden. Somit werden Leerlaufzeiten, die bei aufeinanderfolgenden Speicherzugriffen entstehen, ausgeschaltet.

Befehlseinheit

Die von der Prefetch-Schlange übernommenen Befehle werden vom Befehlsdekoder der Befehlseinheit dekodiert und in einer weiteren Warteschlange, die maximal drei dekodierte Befehle enthält, abgelegt.

Ausführungseinheit

Innerhalb der Ausführungseinheit werden die dekodierten Befehle von der Warteschlange übernommen und bearbeitet. Diese Einheit benutzt die Buseinheit für alle Datenübertragungen vom und zum Speicher bzw. I/O-Port.

Adreßeinheit

Die Adreßeinheit übernimmt das Umrechnen von der virtuellen in die physische Adresse, wie sie von der Buseinheit benötigt wird und überprüft bei jeder Adresse, ob der Zugriff zulässig ist und ob eine Segmentgrenze überschritten wurde.

Registersatz

Die Grundarchitektur des U 80601 enthält 15 Register, die sich in die im folgenden beschriebenen vier Kategorien einteilen lassen.

Allgemeine Register

Acht 16-Bit-Mehrzweckregister enthalten die arithmetischen und logischen Operanden, wobei vier dieser Register (AX, BX, CX, DX) sowohl 16-Bit-Operanden als auch jeweils zwei getrennte 8-Bit-Register enthalten können.

Segmentregister

Vier spezielle 16-Bit-Register ermöglichen den direkten Zugriff auf vier Speichersegmente.

Die Segmentregister dienen zum Identifizieren der vier aktuellen Segmente, wobei jedem Register ein bestimmter Segmenttyp zugeordnet ist.

CS – Codesegment-Register
DS – Datensegment-Register
SS – Stacksegment-Register
ES – Extrasegment-Register

Der Inhalt dieser Register wird als Segmentselektor bezeichnet.

Basis- und Indexregister

Vier der allgemeinen Register können zur Bestimmung der Offsetadressen der Operanden innerhalb des Speichers benutzt werden. Diese Register können die Basisadresse oder den Index (Quellindex, Zielindex) der einzelnen Positionen innerhalb der Segmente enthalten. Die Spezifikation dieser Register, die zur Adreßberechnung der Operanden benutzt werden, ist abhängig vom jeweiligen Adressierungsmodus.

Status- und Steuerregister

Drei spezielle 16-Bit-Register enthalten das Flagwort, den Befehlszeiger (Instruction Pointer) und das Maschinenstatuswort.

Das 16-Bit-Flagwort enthält spezielle Charakteristiken der Ergebnisse logischer und arithmetischer Befehle (Bit 0, 2, 4, 6, 7 und 8) und steuert die Operationen des U 80601 innerhalb der gegebenen Operationsart (Bit 8 bis 10).

Der Befehlszeiger enthält die relative Adresse des als nächstes auszuführenden Befehls innerhalb des aktuellen Segmentes. Gemeinsam mit dem Codesegmentregisters (CS) läßt sich somit ein 32-Bit-Programmzähler ermitteln.

Der Befehlszeiger kann mit Interruptmechanismen, Sprüngen und Steuertransferoperationen implizit kontrolliert werden. Vom 16-Bit-Maschinenstatuswort werden beim U 80601 nur die vier niederwertigen Bits genutzt.

Speicherorganisation

Der Speicher des U 80601 ist als Satz von Segmenten variabler Länge organisiert. Jedes dieser Segmente umfaßt einen linear adressierbaren Speicherbereich von bis zu 64 KByte. Die Adressierung des Speichers erfolgt mit einer Zwei-Komponenten-Adresse. Die Komponenten sind Zeiger, die einen 16-Bit-Segmentselektor und einen 16-Bit-Offset enthalten. Der Segmentselektor weist auf das gewünschte Speichersegment, während die Offsetkomponente die gewünschte Byteadresse innerhalb des Segmentes kennzeichnet. Alle Befehle, die Operanden innerhalb des Speichers adressieren, müssen folglich das Segment und den Offset spezifizieren. Bei schneller und kompakter Befehlsabarbeitung wird der Segmentselektor in der Regel in die *High-speed*-Segmentregister abgelegt. In diesem Falle hat der Befehl nur das gewünschte Segmentregister und den Offset zu spezifizieren, um den Operanden im Speicher zu adressieren. Allerdings ist es bei den meisten Befehlen nicht erforderlich, das benötigte Segmentregister explizit zu spezifizieren, da das richtige Segmentregister (entsprechend festen Regeln) automatisch ausgewählt wird.

Bei segmentüberschreitenden Befehlspräfixen ist es möglich, die impliziten Segmentregisterwahlregeln für spezielle Fälle zu überfahren. Der Stack, die Daten- und die Extra-Segmente können mit denen einfacherer

Bild 1
Blockschaltbild des U 80601

Programme übereinstimmen. Um auf Operanden zugreifen zu können, die nicht in einem der durch die vier Segmentregister unmittelbar verfügbaren Segmente enthalten sind, muß ein vollständiger 32-Bit-Zeiger oder ein neuer Segmentselektor geladen werden.

Ein-/Ausgaberaum

Der I/O-Raum (Eingabe-, Ausgabebereich) des U 80601 besteht aus 64 K 8-Bit-Ports oder 32 K 16-Bit-Ports. I/O-Befehle adressieren den I/O-Raum entweder mit einer 8-Bit-Portadresse, die im Befehl spezifiziert wird oder mit einer 16-Bit-Portadresse des DX-Registers. 8-Bit-Portadressen werden mit 0 ergänzt, so daß die Adressen A_{15}-A_8 Low sind. Die I/O-Portadressen 00F8H bis 00FFH sind reserviert.

Interrupts des U 80601

Ein Interrupt unterbricht die momentane Programmabarbeitung; der Prozessor setzt daraufhin die Abarbeitung an anderer Stelle fort. Die alte Programmadresse (CS:IP) und der Maschinenstatus (Flags) werden in den Stack gerettet und ermöglichen somit die Rückkehr in das unterbrochene Programm. Die Interrupts lassen sich in drei Klassen einteilen:
- Hardwareinterrupts
- INT-Befehle
- Befehlsausnahmen.



Hardwareinterrupts werden in Abhängigkeit vom Zustand eines externen Einganges ausgelöst und können als maskierte oder nicht maskierte Interrupts abgearbeitet werden. Mit einem INT-Befehl kann ein Interrupt aus einem Programm heraus ausgelöst werden. Befehlsaufnahmen werden wirksam, wenn beim Versuch, eine Anweisung abzuarbeiten, unnormale Bedingungen auftreten, die die weitere Befehlsabarbeitung behindern. Die Rückkehradresse einer Ausnahme zeigt immer auf den Befehl, der die Ausnahme hervorgerufen hat und schließt die vorangegangenen Befehlspräfixe mit ein.
In einer Tabelle können bis zu 256 Zeiger definiert werden, die besondere Interrupt-Service-Routinen für jeden Interrupt beinhalten. Die Interrupts 0 bis 31, von denen einige für Befehlsausnahmen genutzt werden, sind bereits vergeben.
Bei jedem Interrupt muß dem U 80601 ein 8-Bit-Interruptvektor, der eine bestimmte Eintrittsmarke kennzeichnet, übergeben werden. INT-Befehle enthalten oder implementieren diesen Vektor und haben Zugriff auf alle 256 Interrupts. Maskierte Hardwareinterrupts liefern der CPU den 8-Bit-Vektor während der *Interrupt Acknowledge Bus Sequence*. Nicht maskierte Hardwareinterrupts verwenden einen intern gelieferten, vorbestimmten Vektor.

Maskierbarer Interrupt (INTR)

Der U 80601 besitzt einen maskierbaren Hardware-Interrupt-Request-Eingang (INTR). Durch Setzen des Interruptflag-Bits (IF) im Flagwort kann dieser Eingang softwaremäßig freigegeben (IF = 1) oder verriegelt (IF = 0) werden. Alle 224 vom Nutzer festlegbaren Interruptquellen können diesen Eingang verwenden, wobei jede dieser Quellen ihre seperate Interruptbehandlung haben kann.
Während der Interrupt-Acknowledge-Sequenz liest die CPU einen 8-Bit-Vektor, mit dem die Interruptquelle identifiziert werden kann. Innerhalb der Bedienung des Interrupts wird das IF-Bit, als Teil der Antwort auf den Interrupt oder die Ausnahme, rückgesetzt, und folglich werden weitere maskierbare Interrupts gesperrt. Das im Stack gerettete Flagwort enthält den Status (des Prozessors), der vor dem Interrupt gültig war. Das Interruptflag bleibt solange auf 0 gesetzt, bis das Flagwort in das Flagregister zurückgeschrieben wird. Der Interrupt-Return-Befehl beinhaltet unter anderem das Rückschreiben des Flagwortes und somit das Rückschreiben des ursprünglichen Status.

Nichtmaskierbarer Interrupt (NMI)

Der U 80601 besitzt weiterhin einen nichtmaskierbaren Interrupteingang (NMI). Der NMI ist höher priorisiert als der INTR. Eine typische Nutzung des NMI ist die Aktivierung einer leistungsfähigen Fehlerroutine. Wird der NMI-Eingang aktiviert, so wird ein Interrupt mit dem intern gelieferten Vektorwert 2 ausgelöst. Eine Interrupt-Acknowledge-Sequenz wird nicht aktiviert.
Während der U 80601 die NMI-Serviceprozedur ausführt, werden keine weiteren NMI-Requests, keine INTR-Requests und keine Interrupt-Requests bei Überschreiten des Koprozessorsegmentes bedient. Falls ein weiterer NMI während einer bereits laufenden NMI-Serviceroutine auftritt, wird dies von der CPU erkannt, und der NMI wird nach der Ausführung der ersten IRET-Anweisung bearbeitet. Das Interruptflag-Bit (IF) wird zu Beginn eines NMI zurückgesetzt (IF = 0), um maskierte Interrupts zu verhindern.

Single-Step-Interrupt

Der U 80601 verfügt über einen internen Interrupt, der es erlaubt, Programme im Einzelschrittbetrieb (Befehl für Befehl) abzuarbeiten. Dieser Interrupt wird als Single-Step-Interrupt bezeichnet und über das Single-Step-Flag-Bit (TF) des Flagwortes gesteuert. Wird dieses Bit einmal gesetzt, wird nach der Ausführung des nächsten Befehls ein Single-Step-Interrupt ausgelöst. Dieser Interrupt nutzt einen intern gelieferten Vektor des Wertes 1 und setzt das TF-Bit zurück. Der IRET-Befehl wird zum Setzen des TF-Bits und zur Steuerung der Single-Step-Abarbeitung des nächsten Befehls genutzt.

Interruptprioritäten

Treten verschiedene Interrupts gleichzeitig auf, so werden sie entsprechend den folgenden Prioritäten abgearbeitet.

Priorität	Interrupttyp
hoch	Befehlsausnahmen
↑	Single Step
	NMI
	Koprozessor-Segment-überschreitung
↓	INTR
niedrig	INT-Befehl

Die Interruptbearbeitung enthält unter anderem das Retten des Stacks und der Rücksprungadresse sowie das Setzen von CS:IP auf den ersten Befehl der Interruptbehandlung.

Real Mode

Der U 80601 arbeitet im *Real Mode* abwärtskompatibel zum Befehlsatz des K 1810 WM86. In dieser Betriebsart ist der U 80601 objektcodekompatibel zu Software, die mit dem Prozessor K 1810 WM 86 erstellt wurde. Der physische Speicher des U 80601 besteht aus einem fortlaufenden Bereich von maximal 1 MByte, der über die Adreßpins A_0 bis A_{19} und mit dem Signal $\overline{BHE}$ adressiert wird. Die Adressen A_{20} bis A_{23} werden ignoriert.
Im Real Mode ist über die Adresse A_0 bis A_{19} ein physischer Speicher von bis zu einem Megabyte adressierbar. Eine Adresse wird hierbei aus zwei Komponenten, einem Segmentselektor und einer Offsetkomponente, gebildet.
Die oberen 16 Bit einer 20-Bit-Segmentadresse werden als Segmentselektor interpretiert. Die vier niederwertigen Bits dieser Segmentadresse sind immer 0. Folglich sind Segmentadressen immer Vielfache von 16 Byte. Die Offsetkomponente wird aus den 16 niederwertigen Bits der Offsetadresse gebildet. Die oberen vier Bit dieser Adresse sind hierbei 0.
Alle Segmente haben im Real Mode einen Umfang von 64 KByte. Ihr Inhalt ist lesbar, beschreibbar und ausführbar. Falls Datenoperanden oder Befehle versuchen, über das Ende eines Segmentes zu greifen (zum Beispiel ein Wort mit dem niederwertigen Offsetbyte FFFFH und dem höherwertigen Offsetbyte 0000H), kann ein Interrupt oder eine Ausnahme ausgelöst werden.
Der U 80601 reserviert im Real Mode zwei feste Speicherzonen, die Systeminitialisierungszone und die Interrupttabellenzone. Speicherplätze von Adresse FFFF0H bis Adresse FFFFFH sind hierbei für die Systeminitialisierung reserviert. Die Ausführung der Initialisierung beginnt folglich auf der Adresse FFFF0H.
Die Speicherplätze von Adresse 00000H bis 003FFH sind für die Interruptvektoren reserviert.

Protected Mode

Der U 80601 arbeitet im *Protected Mode (Virtual-Protected Mode)* abwärtskompatibel zum Befehlssatz des K 1810 WM86. Weiterhin ist im Protected Mode eine leistungsfähige Speicherverwaltung möglich, und es wird ein mit bestimmten Befehlen verbundener Schutzmechanismus wirksam.
Mit dem Setzen des PE-Bits (Protection Enable) des Maschinenstatuswortes mittels LSMW-Befehls (Load Maschine Status Word) gelangt der U 80601 vom Real Mode in den Protected Mode. Der Protected Mode

bietet einen erweiterten physischen und einen virtuellen Speicheradreßraum, einen Speicherschutzmechanismus und neue Operationen, die das Betriebssystem und den virtuellen Speicher unterstützen.
Programme des K1810 WM86 und Programme, die im Real Mode des U80601 erzeugt wurden, sind im Protected Mode ausführbar, allerdings müssen sie in die unterschiedlichen Konstanten der Segmentselektoren eingebettet werden.

Speicherraum und Speicheradressierung

Der U80601 verfügt im Protected Mode über einen virtuellen Adreßraum von 1 GByte pro Task, der in physisch adressierbare Blöcke von je 16 MByte eingeteilt ist und über die Adreßpins A_{23}-A_0 und $\overline{BHE}$ adressiert wird.
Im Protected Mode erfolgt die Adressierung, wie im Real Mode, unter Verwendung eines 32-Bit-Zeigers, der aus einer 16-Bit-Selektorkomponente und einer Offsetkomponente besteht. Der Selektor entspricht in diesem Fall aber nicht den oberen 16 Bit einer realen Speicheradresse, sondern spezifiziert den Index von Tabellen innerhalb des Speichers. Die 24-Bit-Basisadresse des gewünschten Segmentes wird unter Nutzung dieser Tabellen gebildet. Um die physische Adresse zu ermitteln, wird der 16-Bit-Offset zur Segmentbasisadresse addiert. Sobald ein Segmentregister mit einem Selektor geladen wird, werden die Tabellen automatisch von der CPU angesprochen. Alle Befehle des U80601, die Segmentregister laden, greifen ohne zusätzliche Software auf die Tabellen im Speicher zu. Diese Tabellen, Deskriptortabellen genannt, enthalten jeweils 8 Byte.

Deskriptoren

Deskriptoren bestimmen die Verwendung des Speichers. Gatedeskriptoren bestimmen außerdem das Verhalten bei Taskwechseloperationen. Der U80601 besitzt Deskriptoren für Code-, Stack- und Datensegmente und Systemsteuerdeskriptoren für Systemdatensegmente und Steuertransferoperationen. Deskriptorzugriffe werden als gesperrte (locked) Busoperationen ausgeführt, damit die Deskriptoren auch in Multiprozessorsysteme integriert werden können.

Code- und Datensegmentdeskriptoren (S = 1)
Neben der Segmentbasisadresse und der durch die Segmentgrenze festgelegten Segmentlänge enthalten Code- und Datensegmentdeskriptoren ein Zugriffsrechtebyte, das außerdem Attribute des Segmentes enthält. Dieses Byte enthält fünf Informationen.

* Present Bit
Das P-Bit kennzeichnet die Präsenz, das heißt die Verfügbarkeit des Segmentes.
* Deskriptor Privilege Level
Diese zwei Bits enthalten die Privilegstufe des Selektors.
* Accessed Bit
Mit dem A-Bit wird signalisiert, ob auf das Segment bereits zugegriffen wurde.
* Segmentdeskriptor
Bei Code- und Datensegmentdeskriptoren ist das S-Bit stets 1.
* Type
Drei weitere Type-Bits enthalten zusätzliche Informationen, je nachdem ob es sich um einen Code- oder Datensegmentdeskriptor handelt.

Datensegment: Mit E = 0 wird ein Datensegment gekennzeichnet. Das ED-Bit kennzeichnet die Ausdehnungsrichtung des Segmentes entweder in Richtung steigender oder in Richtung fallender Adressen. Dies verdeutlich, ob es sich um einen Stack oder um einen Datenbereich handelt. Ein weiteres Bit (W-Bit) liefert Informationen über die Beschreibbarkeit des Segmentes.
Codesegment: Bei Codesegmenten ist E = 1. Das Conforming-Bit (C-Bit, Anpassungsbit) ist 1. Somit kann das Codesegment nur ausgeführt werden, wenn die momentane Privilegstufe größer oder gleich der Deskriptorprivilegstufe ist und unverändert bleibt.

Systemsegmentdeskriptoren (S = 0, Type = 1–3)
Systemsegmentdeskriptoren sind ähnlich wie Code- und Datensegmentdeskriptoren aufgebaut. Diese Deskriptoren enthalten zusätzliche Systeminformationen. Das P-Bit gibt Auskunft darüber, ob das Segment im physischen Speicher vorhanden ist oder ob es ungültig ist. Das Typefeld spezifiziert ein gültiges Taskstatussegment (Type = 1), eine lokale Deskriptortabelle (Type = 2) oder ein belegtes Taskstatussegment (Type = 3).

Gatedeskriptoren (S = 0, Type = 4–7)
Gates werden zur Steuerung des Zugriffs auf Eintrittspunkte innerhalb eines Zielcodesegmentes benutzt. Je nachdem wie dieser Zugriff erfolgt, unterscheidet man zwischen Callgates, Taskgates, Interruptgates und Trapgates. Die Gates werden im Typefeld spezifiziert. Weiterhin enthalten Gatedeskriptoren den Zielselektor und den Zieloffset.
Die Gates liefern die Stufe des Übergangs eines Steuertransfers von der Quelle zum Ziel. Die Übergänge ermöglichen es der CPU, automatisch Schutztests (Protection Checks) vornehmen zu können und die Eintrittspunkte des Ziels zu kontrollieren. Callgates werden benutzt, um die Privilegstufe zu ändern, Taskgates werden verwendet, um Taskwechsel auszuführen, und Interrupt- und Trapgates ermöglichen das Spezifizieren von Interruptserviceroutinen.
Das Interruptgate kann im Gegensatz zum Trapgate Interrupts sperren (Rücksetzen von IF).

Lokale und globale Deskriptortabellen

Der U80601 verfügt über zwei Deskriptortabellen, die alle Deskriptoren enthalten, die für eine Task zu jedem beliebigen Zeitpunkt erreichbar sind. Eine Deskriptortabelle besteht aus einem linearen Array mit bis zu 8192 Deskriptoren. Die oberen 13 Bit des Selektorwertes enthalten den Index innerhalb einer Deskriptortabelle. Jede Tabelle verfügt über ein 24-Bit-Basisregister, um die Deskriptortabelle im physischen Speicher lokalisieren zu können, sowie ein 16-Bit-Limitregister, das den Deskriptorzugriff auf eine definierte Grenze beschränkt (Bild 2, S. 136).
Eine der Tabellen, die globale Deskriptortabelle (GDT), enthält Deskriptoren, die für alle Tasks verfügbar sind. Die zweite Tabelle, die als lokale Deskriptortabelle (LDT) bezeichnet wird, enthält Deskriptoren, die von den Tasks privat (lokal) benutzt werden können. Jede Task kann ihre eigene lokale Deskriptortabelle besitzen.
Die GDT kann außer Interrupt- und Trapdeskriptoren alle Deskriptortypen enthalten, während die LDT nur Segment-, Taskgate- und Callgatedeskriptoren enthalten kann.
Falls der Segmentdeskriptor zum Zeitpunkt des Zugriffs einer Task auf dieses Segment nicht in einer der beiden Tabellen existiert, kann mit der Task nicht auf dieses Segment zugegriffen werden.
Mit den Befehlen **LGDT** und **LLDT** (LOAD GDT/LDT) wird die Basis und das Limit der globalen und der lokalen Deskriptortabellen geladen. Der LGDT-Befehl und der LLDT-Befehl sind privilegiert, das heißt, sie können nur von berechtigten Programmen, die in der Privilegstufe 0 operieren, ausgeführt werden. Der LGDT-Befehl lädt ein sechs Byte langes Feld, das das 16-Bit-Tabellenlimit und die physische 24-Bit-Basisadresse der globalen Deskriptortabelle enthält. Der LDT-Befehl lädt einen Selektor, der sich auf einen Deskriptor der lokalen Deskriptortabelle bezieht, der wiederum die Basisadresse und das Limit einer LDT enthält.

Interrupt-Deskriptortabelle

Im Protected Mode verfügt der U80601 über eine dritte Deskriptortabelle, die Interrupt-Deskriptortabelle (IDT), über die es möglich ist, 256 Interrupts zu definieren. Diese Tabelle kann nur Taskgates, Interruptgates und Trapgates enthalten. Die Interrupt-Deskriptortabelle verfügt innerhalb der CPU über ein physisches 24-Bit-Basisregister und ein 16-Bit-Limitregister. Der privilegierte LIDT-Befehl lädt diese Register mit einem 6-Byte-Wert, analog zum LGDT-Befehl.
Der Zugriff auf die Interrupt-Deskriptortabelle kann über INT-Befehle, externe Interruptvektoren und Ausnahmen erfolgen, die sich auf diese Tabelle beziehen.
Die IDT muß hintereinander mindestens 256 Byte umfassen, um allen reservierten Interrupts einen Platz zuweisen zu können.

Privilegien

Der U80601 besitzt ein in vier Stufen eingeteiltes Privilegsystem, das die Nutzung privilegierter Befehle sowie den Zugriff auf Deskriptoren und die damit verbundenen Segmente innerhalb einer Task kontrolliert. Dieses 4-Stufen-Privileg stellt eine Erweiterung des User-/Supervisor-Modus, der häufig in Minicomputern zu finden ist, dar.
Die Privilegstufen sind von 0 bis 3 numeriert, wobei die Stufe 0 die höchste Privilegstufe ist. Die Privilegstufen ermöglichen einen Schutz innerhalb der Tasks. Durch die Bereitstellung privater (lokaler) Deskriptortabellen für jede Task werden die Tasks voneinander isoliert. Betriebssystemroutinen, Interruptbehandlungen und andere Systemsoftware können mit jeder Task, die die vier Stufen des Privilegs nutzt, in den virtuellen Adreßraum eingebunden und geschützt werden. Jede Task des Systems besitzt für jede ihrer Privilegstufen einen separaten Stack.
Tasks, Deskriptoren und Selektoren haben ein Privilegstufenattribut, das bestimmt, ob der Deskriptor benutzt werden kann oder nicht. Taskprivilege bewirken die Nutzung von Befehlen und Deskriptoren. Deskriptoren- und Selektorprivilege bewirken nur den Zugriff auf den Deskriptor.

Schutzmechanismus

Der U80601 verfügt über einen umfangreichen Mechanismus zum Schutz vor kritischen Befehlen, die den Ausführungstatus der CPU beeinflussen können (z. B. HLT) und die die Code- oder Datensegmente vor

Tafel 1 ***Anschlußbeschreibung des U 80601 (E – Eingang; A – Ausgang)***

Symbol	Pin	E/A	Funktion
$\overline{BHE}$	1	A	Byte High Enable (low-aktiv) wird aktiviert bei Datenübertragungen auf dem höherwertigen Byte des Datenbusses D_8–D_{15}. 8 Bit breite Systeme, denen normalerweise das höherwertige Datenbyte zugewiesen wird, können mit Hilfe des $\overline{BHE}$-Signals selektiert werden. Das Signal ist hochohmig während Bus Hold Acknowledge. *Übertragungsmöglichkeiten:* $\overline{BHE}$ A_0 Funktion 0 0 Wortübertragung 0 1 Byteübertragung oberes Datenbyte 1 0 Byteübertragung unteres Datenbyte 1 1 nicht benutzt
$\overline{S1}$, $\overline{S0}$	4, 5	A	Bus Status (low-aktiv) signalisiert Aktivitäten des Busses. Gemeinsam mit den Signalen M/$\overline{IO}$ und COD/$\overline{INTA}$ läßt sich der Typ des Buszyklus ermitteln. Signale sind hochohmig während Bus Hold Acknowledge. *Busstatusdefinition:* COD/$\overline{INTA}$ M/$\overline{IO}$ $\overline{S1}$ $\overline{S0}$ Buszyklus 0 0 0 0 Interrupt Acknowledge 0 0 0 1 nicht benutzt 0 0 1 0 nicht benutzt 0 1 0 0 falls A_1 = 1, dann Halt, ansonsten Shutdown 0 1 0 1 Speicherdaten lesen 0 1 1 0 Speicherdaten schreiben 1 0 0 0 nicht benutzt 1 0 0 1 I/O lesen 1 0 1 0 I/O schreiben 1 1 0 0 nicht benutzt 1 1 0 1 Speicheranweisung lesen 1 1 1 0 nicht benutzt x x 1 1 kein Busstatuszyklus
PEREQ $\overline{PEACK}$	6	E A	Processor Extension Operand Request and Acknowledge (PEREQ – high-aktiv), $\overline{PEACK}$ – low-aktiv) erweitert die Speicherverwaltungs- und -schutzmöglichkeiten des U 80601 für den Einsatz von Koprozessoren. Das PEREQ-Eingangssignal veranlaßt den U 80601, eine Datenoperationsübertragung zum Koprozessor zu starten, das $\overline{PEACK}$-Ausgangssignal signalisiert dem Koprozessor die Übertragung des geforderten Operanden. $\overline{PEACK}$ verhält sich asynchron zum Systemtakt. PEREQ ist hochohmig während Bus Hold Acknowledge.
A_{23}–A_0	7–34	A	Address Bus (high-aktiv) Signale zur Ausgabe der physischen und I/O-Port-Adresse. A_0 ist Low bei Datenübertragungen der Pins D_7–D_0. A_{23}–A_{10} sind Low während I/O-Übertragungen. Der Adreßbus ist hochohmig während Bus Hold Acknowledge.
RESET	29	E	System Reset (high-aktiv) setzt die interne Logik des U 80601 zurück. Der U 80601 kann jederzeit, nach einer Low-High-Flanke und einer Aktivierung des RESET-Einganges (Halten auf High) von mindestens 16 Systemtaktperioden, initialisiert werden. Während RESET aktiv ist, sind die Ausgangspins des U 80601 mit folgenden logischen Zuständen belegt: *Pinzustand* *Pinbezeichnung* 1 (High) $\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A_{23}–A_0, $\overline{BHE}$, $\overline{LOCK}$ 0 (Low) M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA hochohmig D_{15}–D_0 Der U 80601 startet seine Operationen nach der High-Low-Flanke an RESET. Diese Flanke muß mit dem Systemtakt synchronisiert werden. Etwa 50 Systemtaktzyklen werden für die interne Initialisierung des U 80601 benötigt, bevor der erste Buszyklus zur Übernahme des Codes von der Power-on-Startadresse erfolgen kann. Eine Low-High-Flanke an RESET, synchronisiert mit dem Systemtakt, beendet einen Prozessorzyklus mit der zweiten High-Low-Flanke des Systemtaktes. Erfolgt die High-Low-Flanke asynchron zum Systemtakt, kann nicht bestimmt werden, welche Phase des Prozessortaktes während der nächsten Systemtaktperiode stattfindet. Systemtaktsynchrone Low-High-Flanken des RESETs sind nur für Systeme erforderlich, bei denen der Prozessortakt phasensynchron zu einem weiteren Takt sein muß.
CLK	31	E	System Clock generiert das grundsätzliche Zeitverhalten für U 80600-Systeme. Der U 80601 teilt diesen Takt zur Erzeugung des Prozessortaktes intern durch zwei. Der durch zwei geteilte Takt kann mit einem externen Taktgenerator durch eine Low-High-Flanke am RESET-Eingang synchronisiert werden.
D_{15}–D_0	36–51	E A	Data Bus (low-aktiv) Dateneingänge während Speicher, I/O- und Interrupt-Acknowledge-Lesezyklen. Datenausgänge während Speicher- und I/O-Schreibzyklen; der Datenbus ist hochohmig während Bus Hold Acknowledge.
$\overline{BUSY}$ $\overline{ERROR}$	53, 54	E E	Processor Extension Busy and Error (low-aktiv) kennzeichnet den Arbeitszustand eines Koprozessors. Ein aktiver $\overline{BUSY}$-Eingang verhindert die Ausführung von WAIT-Befehlen und einigen ESCape-Befehlen, bis $\overline{BUSY}$ wieder inaktiv (High) wird. Der U 80601 kann während seiner Wartezeit auf ein inaktives $\overline{BUSY}$ unterbrochen werden. Ein aktiver $\overline{ERROR}$-Eingang signalisiert Fehler des Koprozessors und veranlaßt den U 80601 bei weiterer Ausführung von WAIT- oder ESCape-Befehlen, einen Interrupt (Interrupt 7) auszulösen.
INTR	57	E	Interrupt Request (high-aktiv) ermöglicht das Unterbrechen eines laufenden Programms und den Sprung in eine Interruptserviceroutine. Interrupt-Requests sind gesperrt, solange das Interrupt-Enable-Bit des Statusregisters auf 0 gesetzt ist. Nachdem der U 80601 einen Interrupt akzeptiert hat, generiert er zwei Interrupt-Acknowledge-Buszyklen zum Lesen eines 8-Bit-Interruptvektors und zum Erkennen der Interruptquelle. Zum Starten eines Interruptprogramms muß INTR bis zum Ende des ersten Interrupt-Acknowledge-Zyklus aktiv bleiben. Der INTR-Eingang wird am Anfang eines jeden Prozessorzyklus abgefragt und muß zwei Prozessortaktperioden vor Ende des gerade aktuellen Befehls High sein, damit der Interrupt vor dem nächsten Befehl akzeptiert wird. Der INTR-Eingang ist pegelsensitiv und kann asynchron zum Systemtakt aktiviert werden.
NMI	59	E	Non-Maskable Interrupt Request (high-aktiv) unterbricht den U 80601 mit der internen Erzeugung eines Interruptvektors (Interrupt 2). Es werden keine Interrupt-Acknowledge-Zyklen ausgelöst. Das Interrupt-Enable-Bit wird nicht ausgewertet. Zum Erkennen eines NMI muß das Signal mindestens vier Taktperioden inaktiv (Low) sein, anschließend auf High gesetzt werden und weitere vier Systemtaktperioden diesen aktiven Zustand halten. Der NMI-Eingang ist flankensensitiv, und der Interrupt kann asynchron zum Systemtakt erfolgen.
$\overline{READY}$	63	E	Bus Ready (low-aktiv) kennzeichnet das Ende eines Buszyklus. Buszyklen können ständig generiert werden, bis sie mit Low des $\overline{READY}$-Signales beendet werden. Während Bus Hold Acknowledge wird das Signal ignoriert.
HOLD HLDA	64	E A	Bus Hold Request and Hold Acknowledge (high-aktiv) steuern die Herrschaft des U 80601-Lokalbusses. Der HOLD-Eingang ermöglicht anderen Mastern, die Verwaltung des Lokalbusses anzufordern. Falls die Busherrschaft an andere Master vergeben werden kann, floaten die Bustreiber in den hochohmigen Zustand, das HLDA-Signal wird aktiviert, und der U 80601 nimmt den Bus-Hold-Acknowledge-Zustand ein. Falls das Hold-Signal inaktiv wird, kann der U 80601 die Lokalbusherrschaft wieder übernehmen, wobei das HLDA-Signal rückgesetzt wird und der Bus-HOLD-Acknowledge-Zustand beendet wird. Die Aktivierung des HOLD-Einganges kann asynchron zum Systemtakt erfolgen.
COD/$\overline{INTA}$	66	A	Code/Interrupt Acknowledge Signal zur Erkennung von Instruction-Fetch-Zyklen bzw. Speicherdatenlesezyklen und Interrupt-Acknowledge-Zyklen bzw. I/O-Zyklen. COD/$\overline{INTA}$ ist hochohmig während Bus Hold Acknowledge.
M/$\overline{IO}$	67	A	Memory I/O Select Signal zur Trennung von Speicher- und I/O-Zyklen. Ist das Signal während des Status-Zyklus (T_s) High, kennzeichnet das einen Speicherzyklus oder einen Halt bzw. Shutdown. Ein Low-Pegel markiert einen I/O-Zyklus oder einen Interrupt-Acknowledge-Zyklus. Das Signal ist hochohmig während Bus Hold Acknowledge.
$\overline{LOCK}$	68	A	Bus Lock (low-aktiv) verhindert, daß andere Systembusmaster innerhalb des folgenden Buszyklus die Busherrschaft übernehmen. Das Signal kann mit Hilfe des LOCK-Befehls aktiviert werden und wird automatisch während der Ausführung des XCHG-Befehls (eXCHanGe – Austausch Register-Speicher oder Register-Akkumulator), innerhalb der Interrupt-Acknowledge-Zyklen (garantiert zwei aufeinanderfolgende Zyklen für INTA) und während der Zugriffe auf Deskriptortabellen aktiviert. $\overline{LOCK}$ ist hochohmig während Bus Hold Acknowledge.
V_{cc}	30, 62	–	Betriebsspannung +5 V
V_{ss}	9, 35, 60	–	Masse
CAP	52	E	Substrat Filter Capacitor Zwischen diesem Pin 52 und Masse ist ein Kondensator 47 nF, ± 20%, 12 V anzuschließen. Dieser Kondensator dient dazu, eventuell auftretende Störspannungsspitzen auszugleichen.

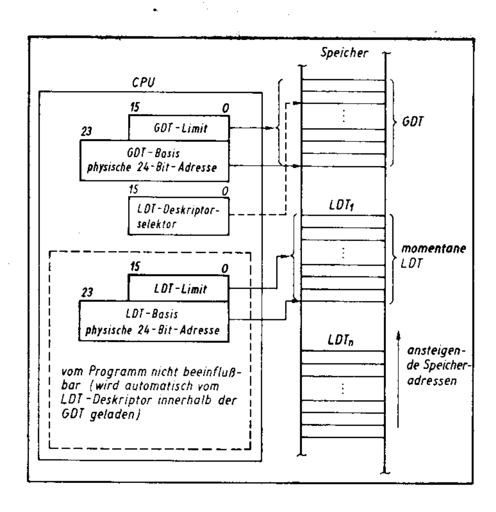


Bild 2 Lokale und globale Deskriptortabellen

ungeeineter Behandlungsweise schützen. Dieser Schutzmechanismus läßt sich in drei Kategorien einteilen:

– eingeschränkte *Behandlung* der Segmente (z. B. Schreibschutz für Datensegmente, die nur lesbar sind); mit den Deskriptoren in der lokalen Deskriptortabelle (LDT) und in der globalen Deskriptortabelle (GDT) können die Segmente in der Gültigkeit ihrer Nutzung festgelegt werden.
– eingeschränkter *Zugriff* auf die Segmente über die Privilegregeln und die Deskriptorbehandlung
– *privilegierte Befehle* oder Operationen, die nur in bestimmten Privilegstufen, die durch die CPL und die I/O-Privilegstufe (IOPL) bestimmt werden, ausgeführt werden können; die IOPL wird durch die Bits 14 und 13 des Flagwortes definiert.

Die mit dem Schutzmechanismus verbundenen Tests (Checks), die bei allen Befehlen ausgeführt werden, lassen sich in drei Gruppen einteilen:
– Segment Load Checks (Segmentladetests)
– Operand Reference Checks (Operandenbezugstests)
– Privileged Instruction Checks (Befehlsprioritätstests).
Wird bei der Ausführung der Tests eine Verletzung der Behandlungsvorschrift festgestellt, so wird eine Ausnahme ausgelöst, die es ermöglicht, auf diese Verletzungen zu reagieren.

Systeminterface

Das Systeminterface des U 80601 kann als Lokalbus und als Systembus ausgeführt werden. Der Lokalbus besteht aus Adreß-, Daten-, Status- und Steuersignalen des U 80601. Ein Systembus ist eine gepufferte Version des Lokalbusses.
Die U 80601-Familie umfaßt verschiedene Bauelemente, mit denen es möglich ist, ein Standardbussystem wie das modulare Mikrorechnersystem MMS 16 zu realisieren.
Das Mikrosystem-Lokalbusinterface spricht lokale Speicher- und I/O-Komponenten des U 80601 an. Dieses Interface besteht aus 24 Adreßleitungen, 16 Datenleitungen und 8 Status- und Steuersignalen.
Ein Systembuskonzept läßt sich beispielsweise mit den Komponenten CPU (U 80601), Taktgenerator (DS 82284), Buscontroller (U 80606), Busarbiter (U 80609) sowie Bustreiber (DS 8286/8287) und Latches (DS 8282/8283) aufbauen.

Der Taktgenerator DS 82284 generiert den Systemtakt und synchronisiert $\overline{\text{READY}}$ und RESET. Der Buscontroller U 80606 wandelt den dekodierten Busoperationsstatus des U 80601 in Kommando- und Bussteuersignale. Der Busarbiter U 80609 generiert MMS 16-busgerechte Signale.
Diese Komponenten liefern das geforderte Zeitverhalten, die elektrischen Pegel und die Treiberleistung für die meisten Systembusinterfaces einschließlich des MMS 16.

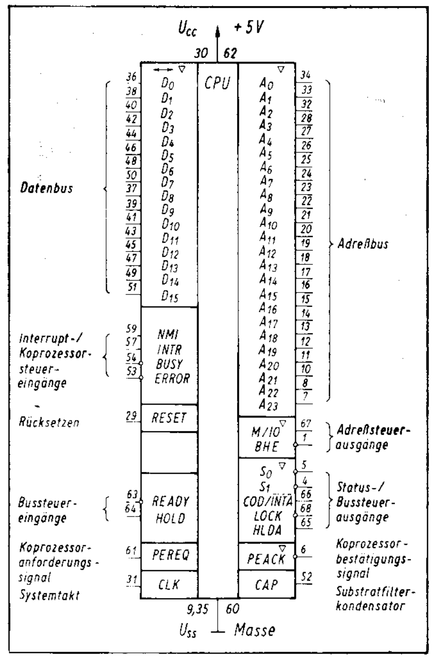


Bild 3 Schaltbild des U 80601

Busoperation

Der U 80601 verwendet einen Systemtakt mit doppelter Frequenz (CLK-Eingang) zur Steuerung des Buszeitverhaltens. Alle Signale des Lokalbusses werden relativ von diesem Systemtakteingang abgeleitet. Zur Erzeugung des internen Prozessortaktes, der den Buszustand bestimmt, wird der Systemtakt durch zwei geteilt. Ein Prozessortakt setzt sich aus zwei Systemtaktzyklen zusammen, die als Phase 1 und Phase 2 bezeichnet werden.
Die CPU unterstützt sechs Typen von Busoperationen:
– Memory Read Operation
– Memory Write Operation
– I/O-Read Operation
– I/O-Write Operation

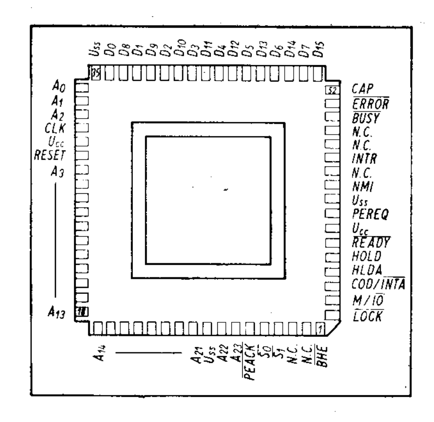


Bild 4 Anschlußbelegung

– Interrupt Acknowledge Operation
– Halt/Shutdown Operation.
Bei der Datenübertragung kann maximal ein Wort innerhalb von zwei Prozessortaktzyklen übertragen werden.
Der U 80601 hat drei grundsätzliche Zustände:
– Idle (T_i)
– Status senden (T_s)
– Kommando ausführen (T_c).
Ein vierter Lokalbuszustand der CPU wird als Halt (T_h) bezeichnet. Der T_h-Zustand zeigt an, daß der U 80601 die Kontrolle des Lokalbusses, als Antwort auf ein HOLD-Request, an einen anderen Busmaster übergeben hat.
Der Idle-Zustand (Ruhezustand) zeigt an, daß kein Datentransfer stattfindet oder angefordert ist. Der erste aktive Zustand T_s wird signalisiert, indem die Statussignale $\overline{S1}$ und/oder $\overline{S0}$ auf Low gesetzt werden. Er kennzeichnet die Phase 1 des Prozessortaktes. Während T_s erfolgt die Dekodierung des Kommandos, und die Adressen und Daten (bei Schreiboperationen) werden freigegeben. Der Buscontroller U 80606 dekodiert die Statussignale und generiert MMS 16-bus-kompatible Read- bzw. Writekommandos sowie die Steuersignale der lokalen Treiber.
Nach dem Zustand T_s wird der Kommandoausführungszustand T_c eingenommen. Speicher- oder I/O-Systeme reagieren auf diese Busoperation innerhalb von T_c, indem sie entweder Lesedaten zur CPU übertragen oder Schreibdaten akzeptieren. Um die Reaktionszeit von Speicher- oder I/O-Systemen genügend abzusichern, werden T_c-Zustände so oft als notwendig wiederholt. Das READY-Signal bestimmt, ob T_c wiederholt wird oder nicht.
Ein wiederholter T_c-Zustand wird als WAIT-Zustand bezeichnet. Während des Haltzustandes (T_h) sind alle Adreß-, Daten- und Statusausgangspins des U 80601 hochohmig, um einem anderen Busmaster die Nutzung des lokalen Busses zu erlauben. Das HOLD-Eingangssignal wird verwendet, um den U 80601 in den T_h-Zustand zu setzen. Das HLDA-Ausgangssignal signalisiert, daß die CPU den T_h-Zustand eingenommen hat.

Zusammenfassung

Der U 80601 ist ein schneller 16-Bit-Mikroprozessor, der für den Einsatz in leistungsfähigen Personalcomputern und automatischen Steuerungen entwickelt wurde. Hierbei erfüllt der U 80601 internationale PC-Standardforderungen.
Der U 80601 zeichnet sich durch eine hohe Arbeitsgeschwindigkeit (Taktfrequenz bis zu 16 MHz) bei erhöhtem Datendurchsatz (Pipelining, Prefetching) und einen großen Adreßbereich (ein Gigabyte pro Task) aus.
Zur Absicherung der Entwicklung und der Produktion einer neuen, leistungsfähigen Personalcomputergeneration wird mit dem U 80601 eine CPU bereitgestellt, die neuesten internationalen Anforderungen entspricht (Einsatzmöglichkeit in Forschung, Entwicklung, Konstruktion, Prozeßüberwachung u. a.).

⊠ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76 App. 25

Der Buscontroller U 80606 DC

Volkmar Heilbock
VEB Mikroelektronik „Karl Marx“
Erfurt, Forschungszentrum

Vorbemerkung

Der im VEB Mikroelektronik „Karl Marx“ Erfurt gefertigte integrierte Schaltkreis U 80606 DC ist ein Buscontroller für den schnellen 16-Bit-Mikroprozessor U 80601. Er stellt den angrenzenden Bussystemen die wichtigsten Befehls- und Steuersignale zur Verfügung und steigert damit die Leistungsfähigkeit der CPU erheblich.

Der U 80606 DC wird im U 80600-System zur Erzeugung der Adreßlatch-Steuersignale, zur Steuerung der Datenübertragung und zur Standard-Befehlsausgabe verwendet. Die Befehlsausgaben erfolgen zeitgesteuert und erfüllen alle Forderungen von MMS 16-Bussystemen (Multibus). Die Bilder 1 und 2 zeigen die interne Struktur und das Schaltbild des U 80606 DC.

Mittels fester Pinbeschaltung können beim U 80606 DC zwei Betriebsarten für das jeweilige Buszeitverhalten eingestellt werden, für

- MMS 16-kompatible Buszyklen
- schnelle lokale Buszyklen.

Der Buscontroller verfügt über

- einen Takteingang (U 80601-Systemtakt)
- acht Status- und Steuereingänge
- fünf Befehlsausgänge
- vier Steuerausgänge.

Der U 80606 DC liefert an den Steuersignalausgängen 16 mA und an den Befehlsausgängen 32 mA Treiberstrom bei Low-Pegel. Für Speicher- und I/O-Baugruppen stehen jeweils getrennte Befehlsausgänge zur Verfügung. Der Datenbus wird von separaten Datenfreigabesignalen und von Signalen zur Festlegung der Übertragungsrichtung gesteuert.

Der U 80606 DC wird im 20poligen Dual-in-line-Plastgehäuse geliefert.

Anschluß- und Funktionsbeschreibung

Bild 3 zeigt die Anschlußbelegung des U 80606 DC. In der Anschlußbeschreibung in Tafel 1 sind auch ausführliche Hinweise zur Funktion und Handhabung des U 80606 DC enthalten.

Ergänzungen und Definitionen

Für den Begriff *CPU* steht der Prozessor U 80601 oder ein Zusatzschaltkreis des U 80600-Systems, der die Funktion eines lokalen Busmasters erfüllen kann und somit die Statuseingänge des U 80606 DC treibt.

Prozessorzyklus

Jede CPU, die einen lokalen Bus treibt, arbeitet mit einem internen Takt, dessen Frequenz der halben Taktfrequenz des Systemtaktes entspricht. Die Kenntnis der Phase des internen Taktes des Busmasters ist für die richtige Funktion des lokalen U 80600-Busses erforderlich. Der Busmaster informiert den Buscontroller über die Phasenlage seines internen Taktes, indem er die Statussignale ausgibt. Die Statussignale belegen immer die Phase 1 des internen Taktes des Busmasters.

Busstatus

Der Buscontroller U 80606 DC erlaubt drei Arten des Buszustandes (Busstatus): Idle (T_i), Status (T_s) und Befehl (T_c). Jeder Busstatus ist zwei Systemtaktzyklen lang. Die Phasen innerhalb der Buszustände korrespondieren mit den Phasen des internen CPU-Taktes.

Der T_i-Zustand wird eingenommen, wenn gerade kein Buszyklus auf dem lokalen U 80600-Bus aktiv ist. Dieser Zustand kann mit unbestimmter Häufigkeit wiederholt werden. In der Zeit, in der gerade die Steuerung des lokalen Busses von einem Master an einen anderen Master übergeben wird, verbleibt der Bus ebenfalls im T_i-Zustand.

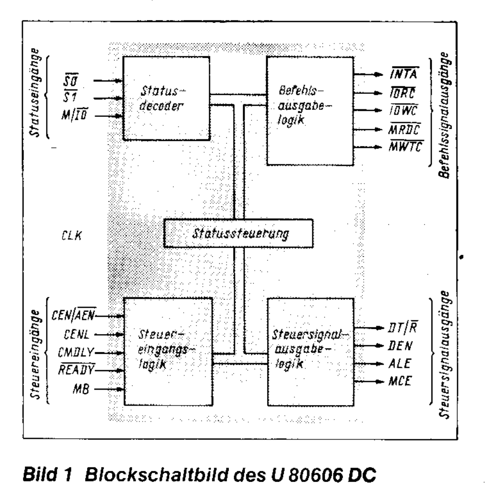


Bild 1 Blockschaltbild des U 80606 DC

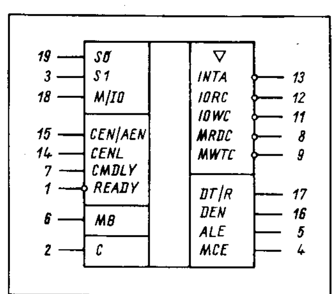
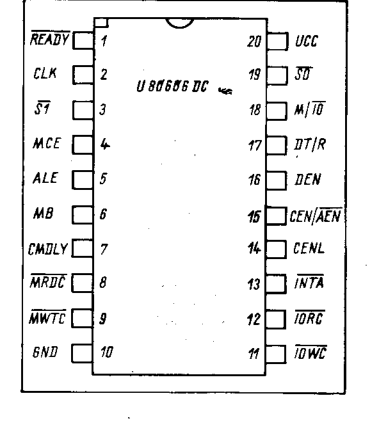


Bild 2 Schaltbild des U 80606 DC

Bild 3 Anschlußbelegung des U 80606 DC ▶



Tafel 1 Anschlußbeschreibung des U 80606 (E – Eingang; A – Ausgang)

Symbol	Pin	E/A	Funktion
CLK	2	E	Clock Aus dem Systemtakt wird das gesamte Zeitverhalten des U 80600-Systems abgeleitet. Das Abfragen der Eingänge des U 80606 DC und ein Wechsel der Pegel an dessen Befehls- und Steuersignalausgängen wird von einer fallenden Taktflanke ausgelöst.
$\overline{S0}$ $\overline{S1}$	19 3	E E	Status 0 (low-aktiv) Status 1 (low-aktiv) Diese beiden Signale bestimmen in Verbindung mit dem M/$\overline{IO}$-Eingang den Typ des jeweiligen Buszyklus. Wenn mit einer fallenden CLK-Flanke entweder an $\overline{S0}$ oder an $\overline{S1}$ Low-Pegel erkannt wurde, dann beginnt ein Buszyklus.
M/$\overline{IO}$	18	E	Memory/IO Dieses Signal legt fest, ob der aktuelle Buszyklus im Speicherbereich oder im I/O-Bereich abläuft. Bei Low-Pegel an M/$\overline{IO}$ findet der Buszyklus im I/O-Bereich statt.

M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Buszyklustyp	aktiver Befehl	DT/R	ALE, DEN?	MCE?
0	0	0	Interrupt-Acknowledge	INTA	0	ja	ja
0	0	1	I/O-Lesen	IORC	0	ja	nein
0	1	0	I/O-Schreiben	IOWC	1	ja	nein
0	1	1	Idle State	–	1	nein	nein
1	0	0	HALT/Shutdown	–	1	nein	nein
1	0	1	Speicher-Lesen	MRDC	0	ja	nein
1	1	0	Speicher-Schreiben	MWTC	1	ja	nein
1	1	1	Idle State	–	1	nein	nein

Symbol	Pin	E/A	Funktion
MB	6	E	Multibus Mode Mit diesem Eingang wird das Zeitverhalten der Steuer- und der Befehlsausgänge gesteuert. Bei High-Pegel arbeitet der Buscontroller mit multibus-kompatiblem Zeitverhalten. Bei Low-Pegel optimiert der Buscontroller das Zeitverhalten dieser Ausgänge für kurze Buszyklen. Mit dem MB-Signal wird die Bedeutung des Eingangs CEN/$\overline{AEN}$ festgelegt. Der MB-Eingang wird im allgemeinen fest verdrahtet und nicht dynamisch betrieben.
CENL	14	E	Command Enable Latched Dieses Signal gibt den Buscontroller entsprechend dem aktuellen Buszyklus frei. CENL ist high-aktiv und wird intern am Ende eines jeden Buszyklus gelatcht. Der Eingang dient der Auswahl des passenden Buscontrollers für jeden Buszyklus, wenn die CPU mehr als einen Bus (d. h. auch mehr als einen Buscontroller) nutzt. CENL wird mit U_{cc} verbunden, wenn der Buscontroller für alle Zyklen verfügbar sein soll.
CMDLY	7	E	Command Delay Dieses Signal verzögert die Signalausgabe an den Befehlsausgängen. Wenn bei der internen Abfrage des CMDLY-Eingangs High-Pegel anliegt, wird der Befehls-

Symbol	Pin	E/A	Funktion
			ausgang nicht aktiviert und CMDLY im nächsten Taktzyklus erneut abgefragt. Bei Low-Pegel wird der selektierte Befehl freigegeben. Wenn am $\overline{READY}$-Eingang *vor* Aktivierung des Befehlsausgangs Low-Pegel erkannt wurde, beendet der Buscontroller den Buszyklus auch ohne Ausgabe eines Befehls. CMDLY wird mit GND verbunden, wenn keine Verzögerungen vor der Ausführung eines Befehls nötig sind. Die Steuersignalausgänge werden von CMDLY nicht beeinflußt.
$\overline{READY}$	1	E	Ready (low-aktiv) Über diesen Eingang wird dem Buscontroller das Ende des aktuellen Buszyklus signalisiert. Der Multibus Mode erfordert mindestens einen WAIT-Zustand, um die Befehlsausgänge aktiv werden zu lassen. Während RESET muß $\overline{READY}$ auf Low-Pegel liegen, um den U 80606 DC in den Idle State (kein Buszyklus) zu bringen.
CEN/$\overline{AEN}$	15	E	Command Enable/Address Enable Hiermit werden die Befehlsausgänge und der DEN-Ausgang des Buscontrollers gesteuert. CEN/$\overline{AEN}$ kann asynchron zum Takt gesetzt werden. Der Eingang kann mit U_{cc} oder GND verbunden werden. Wenn am MB-Eingang High-Pegel anliegt, gilt die $\overline{AEN}$-Funktion. Low-Pegel an $\overline{AEN}$ zeigt an, daß der CPU der Bus überlassen wurde – der Buscontroller beendet dann den hochohmigen Zustand der Befehlsausgänge und bringt sie in den inaktiven Zustand (High). Bei High-Pegel an $\overline{AEN}$ hat die CPU keinen Zugriff auf den Bus, die Befehlsausgänge sind hochohmig und DEN wird inaktiv (Low). $\overline{AEN}$ wird im allgemeinen von einem Busarbiter gesteuert. Bei Low-Pegel an MB hat der Eingang CEN-Funktion. CEN ist ein ungelatchter high-aktiver Eingang zur Aktivierung der Befehlsausgänge und des DEN-Ausgangs. Bei Low-Pegel an MB *und* an CEN werden die Befehlsausgänge und DEN inaktiv, die Befehlsausgänge werden nicht hochohmig.
ALE	5	A	Address Latch Enable Durch die Ansteuerung der Adreßlatches werden die Adressen während eines Buszyklus stabil gehalten. Während HALT-Buszyklen wird ALE nicht ausgegeben. Von den Steuereingängen wird ALE nicht beeinflußt.
MCE	4	A	Master Cascade Enable Dieser Ausgang zeigt an, daß eine Kaskade-Adresse von einem Master-Interruptcontroller auf den CPU-Adreßbus gelegt und über ALE-Steuerung in die Adreßlatches übernommen werden soll. Der CPU-Adreßbus kann dann die Kaskade-Adresse zu den Slave-Interruptcontrollern senden, so daß einer von ihnen auf den Interrupt-Acknowledge-Zyklus antwortet. MCE ist nur während Interrupt-Acknowledge-Zyklen aktiv (High-Pegel) und wird nicht von den Steuersignaleingängen beeinflußt. Die Nutzung von MCE für die Freigabe der Kaskaden-Adreßtreiber erfordert die Verwendung von Latches, die die Kaskade-Adresse mit der fallenden Flanke von ALE speichern.
DEN	16	A	Data Enable Dieser Ausgang gibt die Datenübertragung auf den lokalen Datenbus frei. Im Multibus Mode wird DEN bei Schreibzyklen verzögert.
DT/$\overline{R}$	17	A	Data Transmit/Receive Dieses Signal bestimmt die Richtung der Datenbewegung. High-Pegel zeigt an, daß gerade ein Schreibzyklus ausgeführt wird, Low-Pegel signalisiert einen Lesezyklus. Wenn DT/$\overline{R}$ den Status wechselt, ist DEN inaktiv. Wenn kein Buszyklus läuft, führt DT/$\overline{R}$ High-Pegel. Von den Steuereingängen wird DT/$\overline{R}$ nicht beeinflußt.
$\overline{IOWC}$	11	A	I/O Write Command (low-aktiv) Dieses Signal veranlaßt eine I/O-Baugruppe, Daten vom Datenbus zu lesen. Der Zeitpunkt der Aktivierung dieses Ausgangs wird von MB und CMDLY gesteuert. Mit $\overline{READY}$ wird $\overline{IOWC}$ inaktiv.
$\overline{IORC}$	12	A	I/O Read Command (low-aktiv) Dieses Signal veranlaßt eine I/O-Baugruppe, Daten auf den Datenbus zu legen. Der Zeitpunkt der Aktivierung dieses Ausgangs wird von MB und CMDLY gesteuert. Mit $\overline{READY}$ wird $\overline{IORC}$ inaktiv.
$\overline{MWTC}$	9	A	Memory Write Command (low-aktiv) Dieses Signal veranlaßt eine Speicherbaugruppe, Daten vom Datenbus zu lesen. Der Zeitpunkt der Aktivierung dieses Ausgangs wird von MB und CMDLY gesteuert. Mit $\overline{READY}$ wird $\overline{MWTC}$ inaktiv.
$\overline{MRDC}$	8	A	Memory Read Command (low-aktiv) Diese Signal veranlaßt eine Speicherbaugruppe, Daten auf den Datenbus, zu legen. Der Zeitpunkt der Aktivierung dieses Ausgangs wird von MB und CMDLY gesteuert. Mit $\overline{READY}$ wird $\overline{MRDC}$ inaktiv.
$\overline{INTA}$	13	A	Interrupt Acknowledge (low-aktiv) Dieses Signal teilt einer interruptanfordernden Baugruppe die Bestätigung dieser Interruptanforderung mit. Der Zeitpunkt der Aktivierung dieses Ausgangs wird von MB und CMDLY gesteuert. Mit $\overline{READY}$ wird $\overline{INTA}$ inaktiv.
U_{cc}	20		Betriebsspannung +5 V
GND	10		Masse

✉ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Applikationszentrum Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76, App. 25

Schaltkreis zur Fehlererkennung und -korrektur U 80608

Rainer Franke, Frank Meinecke
VEB Mikroelektronik „Karl Marx"
Erfurt, Forschungszentrum

Einführung

In Computersystemen ist in den letzten Jahren die Hauptspeicherkapazität und die Anzahl der verwendeten Speicherbauelemente, insbesondere die der dynamischen RAMs, stark angewachsen. Die Zuverlässigkeit der Computersysteme wird damit zunehmend von der Zuverlässigkeit der Speichersysteme bestimmt. Die Wahrscheinlichkeit eines Ausfalls wird in FIT (failure in time) angegeben, wobei 1 FIT einem Ausfall in einer Milliarde Bauelementestunden entspricht. Bei DRAMs sind vor allem zwei Fehlermechanismen zu beachten.

Erstens hat ein DRAM wie jedes andere Bauelement eine endliche Lebensdauer. Die durch diese Fehler (Hard Errors) bedingten Ausfälle von DRAMs betragen heute etwa 50 bis 60 FIT. Zweitens kann besonders bei dynamischen Speichern ein Informationsverlust durch eine externe Störung eintreten. Die Information eines Bits (in einem dynamischen Speicherschaltkreis) ist in der Ladung eines Kondensators mit einer Kapazität von etwa 50 fF (Femtofarad) gespeichert. Durch äußere Einflüsse, wie Störspannungen, aus dem Gehäusematerial entstehende Alphastrahlung oder hochenergetische Höhenstrahlung, kann aber der Kondensator trotz korrekt arbeitender Refresh-Steuerung entladen werden. Diese Fehler führen nicht zu einer Zerstörung des Bauelementes, sondern nur zu einem Informationsverlust. Die Wahrscheinlichkeit für Fehler dieser Art (Soft Errors) liegt bei etwa 500 bis 1000 FIT. Bei einer Speicherkapazität von 64 KByte fällt das Problem nicht ins Gewicht. Die mittlere Zeit zwischen zwei Fehlern (mean time between failures = MTBF) liegt hier für Hard Errors bei 204 Jahren und für Soft Errors noch über 14 Jahren. Anders ist das bei Speichern mit einer Kapazität von mehreren Megabyte, wie sie in Mini- und Mikrorechnern verwendet werden. Bei einer Speicherkapazität von 4 MByte, aufgebaut mit 512 Stück 64-KBit-RAMs sinkt die MTBF bei Hard Errors auf 3,4 Jahre und bei Soft Errors auf 0,22 Jahre. Solche Fehlerraten sind in vielen Einsatzfällen nicht akzeptabel.

Da es sich bei den meisten Fehlern, wie oben gezeigt, um Soft Errors, also um einen Informationsverlust, und nicht um einen Defekt des Bauelementes handelt, kann durch entsprechende Systemkonzepte, bei denen ein fehlerkorrigierender Code im Speicher verwendet wird, das Problem weitgehend behoben werden.

Der VEB Mikroelektronik Erfurt fertigt für diese Anwendungen einen Schaltkreis zur Fehlererkennung und -korrektur (Error Detection and Correction = EDC), der in einem solchen System einen fehlerkorrigierenden Code erzeugt und im Fehlerfall die entsprechende Korrektur vornimmt.

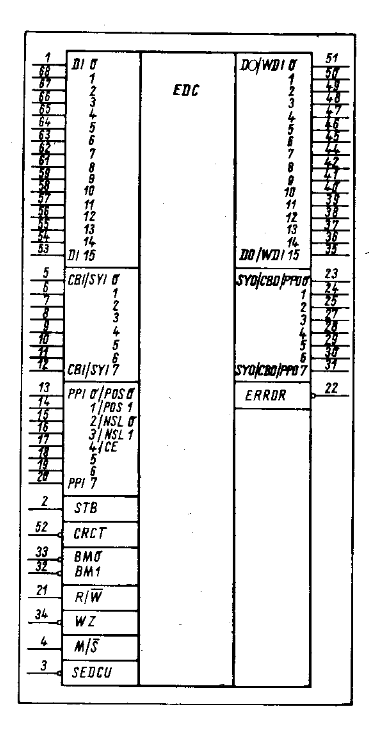


Bild 1 Schaltbild des U 80608

Kurzcharakteristik und Pinbeschreibung

Der EDC U 80608 ist ein sehr schneller Schaltkreis für die Fehlererkennung und -korrektur in statischen und dynamischen Speichersystemen, die hohe Zuverlässigkeit und Leistung erfordern. Er weist folgende Merkmale auf:

- erkennt und korrigiert alle Ein-Bit-Fehler
- erkennt alle Zwei-Bit-Fehler und manche Mehr-Bit-Fehler
- benötigt maximal 52 ns für Fehlererkennung und maximal 67 ns für Fehlerkorrektur (16-Bit-System)
- Syndromausgänge für Fehlerlokalisierung
- getrennter Ein- und Ausgabebus, kein Systemtakt erforderlich
- kaskadierbar bis 80 Bit Speicherbreite
- U 80608–2 ist nicht kaskadierbar. Sein Timing entspricht den Erfordernissen von 16-Bit-CPUs.
- ermöglicht folgende Speicherzugriffsarten:
  – Speicherlesen mit/ohne Fehlerkorrektur (Read)
  – Speicherschreiben (Write)
  – Byte-Schreiben (8 Bit; Partial Write)
  – Lesen-Ändern-Schreiben (Read-Modify-Write)
- 68-Pin-LCCC-Gehäuse
- +5-V-Spannungsversorgung.

Bild 1 zeigt das Schaltbild des U 80608, und Tafel 1 erklärt die Pins des Schaltkreises.

Funktionelle Beschreibung

Der U 80608 ist ein Einchip-System, das alle Ein-Bit-Fehler erkennt und korrigiert sowie alle Zwei-Bit-Fehler und manche Mehr-Bit-Fehler erkennt. Einige ungerade Mehr-Bit-Fehler (z. B. 5 falsche Bits) werden als Ein-Bit-Fehler interpretiert. Einige gerade Mehr-Bit-Fehler (z. B. 4 falsche Bits) können nicht als Fehler erkannt werden. Die meisten Fehler werden als Zwei-Bit-Fehler ermittelt. Die Fehlerbehandlung ist direkt abhängig von der Anzahl der verwendeten Checkbits und dem verwendeten spezifischen Hammingcode. Es wird nicht zwischen Fehlern in den Checkbits und Fehlern in den Datenbits unterschieden.

Jeder U 80608 kann 8 oder 16 Datenbits und bis zu 8 Prüfbits verarbeiten. Durch Kaskadierung von maximal 5 Schaltkreisen U 80608 können Datenwörter mit einer Breite von bis zu 80 Bit verarbeitet werden. Andere Möglichkeiten des U 80608 sind Speicherinitialisierung, Lokalisierung von Speicherfehlern und Byte-Schreiben (8 Bit).

Tafel 1 Anschlußbeschreibung U 80608 (E – Eingang; A – Ausgang)

Symbol	Pin	E/A	Funktion
DI 0–15	1, 68–61, 59–53	E	Data In Eingänge für ein 16-Bit-Datenwort vom RAM für Fehlererkennung und/oder -korrektur
CBI/SYI0 CBI/SYI1 CBI/SYI2 CBI/SYI3 CBI/SYI4 CBI/SYI5 CBI/SYI6 CBI/SYI7	5 6 7 8 9 10 11 12	E E E E E E E E	Check Bits In/Syndrome In In einem Sigle-U 80608-System oder für den Master in einem Multi-U 80608-System sind es Eingänge für 5–8 Checkbits vom RAM. In einem Single-16-Bit-U 80608-System werden CBI0-5 genutzt. Als Slave empfängt der U 80608 die Syndrombits beim Master.
DO/WDI 0 DO/WDI 1 DO/WDI 2 DO/WDI 3 DO/WDI 4 DO/WDI 5 DO/WDI 6 DO/WDI 7 DO/WDI 8 DO/WDI 9 DO/WDI 10 DO/WDI 11 DO/WDI 12 DO/WDI 13 DO/WDI 14 DO/WDI 15	51 50 49 48 47 46 45 44 42 41 40 39 38 37 36 35	E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A E/A	Data Out/Write Data In Bei einem Lesezyklus liegen an diesen Ausgängen die korrigierten ($\overline{\mathrm{CRCT}}$ = Low) oder unkorrigierten ($\overline{\mathrm{CRCT}}$ = High) Daten von DI0-DI15 an. Die $\overline{\mathrm{BM}}$-Eingänge müssen High sein, um die Ausgangstreiber während eines Lesezyklus zu aktivieren. Beim Schreibzyklus liegen an diesen Eingängen zum Berechnen der Checkbits die zu schreibenden Daten an. Ist beim Byte-Schreiben $\overline{\mathrm{BM0}}$ = High, so wird das Byte DO0–DO7 nicht verändert. Dasselbe trifft für das Byte DO8–DO15 bei $\overline{\mathrm{BM1}}$ = High zu. Ist $\overline{\mathrm{WZ}}$ = Low, sind alle Ausgänge DO0–DO15 Low, und die zugehörigen Schreib-Checkbits erscheinen an CBO.
SYO/CBO PPO0 1 2 3 4 5 6 7	 23 24 25 27 28 29 30 31	 A A A A A A A A	Syndrome Out/Check Bits Out/Partial Parity Out al Parity Out In einem Single-U 80608-System oder vom Master eines Multi-U 80608-Systems werden beim Lesezyklus die Syndrombits ausgegeben. Beim Schreibzyklus erscheinen die Checkbits. Als Slave liefert der U 80608 die Teilparität, die der Master während des Read-Modify-Write-Zyklus benutzt. Wenn R/$\overline{\mathrm{W}}$ auf Low geht, werden (beim RMW-Zyklus) die Syndrome gespeichert.
PPI0/POS0 PPI1/POS1	13 14	E E	Partial Parity In/Position Der Master in einem Multi-U 80608-System übernimmt an diesen Eingängen die Paritätsbits 0 und 1 von den Slaves. Einem Slave wird hierüber seine Position im System (1–4) mitgeteilt. Im Single-System nicht verwendet. Pins verfügen über interne Pull-up-Widerstände.
PPI2/NSL0 PPI3/NSL1	15 16	E E	Partial Parity In/Number of Slaves Der Master in einem Multi-U 80608-System übernimmt an diesen Eingängen die Paritätsbits 2 und 3 von den Slaves. Dem Slave 1 wird hierüber die Gesamtzahl der Slaves im System (1–4) mitgeteilt. In einem Single-System und bei den anderen Slaves nicht verwendet. Pins verfügen über interne Pull-up-Widerstände.
PPI4/CE	17	E/A	Partial Parity In/Correctable Error Der Master in einem Multi-U 80608-System übernimmt hierüber das Paritätsbit 4. Beim Slave 1 oder in einem Single-System ist es der Flagausgang für korrigierbare Fehler. CE wird gespeichert, wenn R/$\overline{\mathrm{W}}$ Low-Pegel annimmt. Wird bei Slaves 2–4 nicht verwendet.
PPI5 PPI6 PPI7	18 19 20	E E E	Partial Parity In Der Master übernimmt an diesen Pins die Paritätsbits 5–7 der Slaves. Die Anzahl der Paritätsbits ist gleich der Anzahl der Checkbits. Werden in Slaves und in einem Single-System nicht verwendet
$\overline{\mathrm{ERROR}}$	22	A	Error (low-aktiv) Gibt in einem Single-System oder beim Master in einem Multi-U 80608-System das Error-Flag aus. Wird bei Slaves nicht verwendet. Wird gespeichert, wenn R/$\overline{\mathrm{W}}$ auf Low geht.
$\overline{\mathrm{CRCT}}$	52	E	Correct (low-aktiv) Ist dieses Pin Low, wird die Korrektur der Daten während eines Read- oder Read-Modify-Write-Zyklus durchgeführt. Ist es High, wird die Korrektur ausgeschaltet, die Fehlererkennung ist aktiv.
STB	2	E	Strobe Steuert die Speicherung der Daten an DI und der Checkbits an CBI/SYI. Die Daten werden mit der H/L-Flanke übernommen. Ist das Signal High, werden die Daten an den Eingängen durchgeschaltet.
$\overline{\mathrm{BM0}}$ $\overline{\mathrm{BM1}}$	33 32	E E	Byte Marks (low-aktiv) High-Pegel: DO-Pins sind freigegeben für Lesezyklus. Low-Pegel: DO Pins sind hochohmig für Schreibzyklus. $\overline{\mathrm{BM0}}$ steuert DO0–DO7, $\overline{\mathrm{BM1}}$ steuert DO8–DO15. Beim Byte-Schreiben (8 Bit) ist BM-Eingang Low für neu zu schreibendes Byte.
R/$\overline{\mathrm{W}}$	21	E	Read/Write High-Pegel verlangt vom U 80608 Fehlererkennung und -korrektur, wenn $\overline{\mathrm{CRCT}}$ = Low ist. Low-Pegel erlaubt das Generieren der Checkbits. Mit H/L-Flanke werden die Syndrombits intern für den Read-Modifiy-Write-Zyklus gespeichert
$\overline{\mathrm{WZ}}$	34	E	Write Zero (low-aktiv) Verwendet für Speicherinitialisierung. Low-Pegel: Alle Ausgänge DO0–DO15 sind Low und die entsprechenden Checkbits liegen an CB0–CB7 an. $\overline{\mathrm{BM0}}$, $\overline{\mathrm{BM1}}$ und R/$\overline{\mathrm{W}}$ werden ignoriert.
M/$\overline{\mathrm{S}}$	4	E	Master/Slave High-Pegel: Master Low-Pegel: Slave
$\overline{\mathrm{SEDCU}}$	3	E	Single EDC Unit (low-aktiv) Low-Pegel: einzelner U 80608, High-Pegel: Master im Multi-U 80608-System. Nicht verwendet bei Slaves.
U_{cc}	60	E	Betriebsspannung +5 V
U_{ss}	26	E	Masse für Logik
U_{ss}	43	E	Masse für Ausgänge

Der U 80608 hat eine Flow-Through-Architektur, das heißt, er arbeitet nicht sequentiell. Er unterstützt zwei Arten der Fehlerbehandlung:

① *Flow Through* oder *Correct Always* (Fehlererkennung und -korrektur)
② *Parallel* oder *Check Only* (nur Fehlererkennung)

Es gibt zwei separate 16-Bit-Busse. Der erste übernimmt die Daten vom RAM (DI), und der zweite übergibt die korrigierten Daten an den Systembus (DO/WDI). Die interne Logik ist während eines Lesezyklus vollständig kombinatorisch. Das ist der Unterschied zur Architektur eines Systems, bei dem zuerst Daten gelesen und anschließend die korrigierten Daten geschrieben werden müssen. (Diese Architektur benötigt überlicherweise zusätzliche Hardware, zum Beispiel Latches und bidirektionale Datenbustreiber, und ist durch das erforderliche Zeitverhalten der Steuersignale langsamer.)

Lesezyklus

Wenn das R/$\overline{W}$-Pin High ist, übernimmt der U 80608 die Daten der RAM-Ausgänge über die DI-Eingänge. Wenn es erforderlich ist, können die Daten mit dem STB-Signal zwischengespeichert werden. Es werden die Checkbits des Datenwortes ermittelt und mit den Checkbits des Checkbitspeichers, die an den CBI-Pins anliegen, verglichen. Bei einem Unterschied wird das $\overline{\text{ERROR}}$-Flag aktiv (Low). Das System wird durch das CE-Flag informiert, ob der Fehler korrigierbar oder nicht korrigierbar war. Sind die $\overline{\text{BM}}$-Eingänge High und der Fehler korrigierbar, erscheint das Datenwort korrigiert an den DO-Ausgängen. Ist der Fehler nicht korrigierbar, wird das fehlerhafte Datenwort an die DO-Ausgänge weitergegeben.
Werden mehrere U 80608 kaskadiert, liest der Master die Checkbits. Die Slaves generieren eine Teilparität (Partial Parity) ihres zugehörigen Teilwortes und übergeben sie über die PPO-Ausgänge dem Master. Dieser erzeugt nun das Syndromwort und übergibt es über die SYO-Ausgänge an die Slaves, die nun ihre Daten korrigieren können.
Hat das $\overline{\text{CRCT}}$-Pin High-Pegel, arbeitet der U 80608 in einem Modus, bei dem die Daten nur auf Fehler überprüft, aber nicht korrigiert werden (Check Only Mode). Die Verzögerung zwischen Speicherausgabe und U 80608-Ausgabe ist in diesem Modus bedeutend kürzer als im Korrekturmodus. Im Korrekturmodus signalisiert das $\overline{\text{ERROR}}$-Pin der CPU, daß ein Fehler vorliegt. Die CPU kann dann verschiedene Operationen ausführen, zum Beispiel den laufenden Zyklus zum Zweck der Korrektur verlängern, den Befehl wiederholen oder eine Diagnoseroutine starten.
Das Syndromwort mit einer Breite von 5 bis 8 Bit, das alle notwendigen Informationen über die Existenz und die Bitposition eines Fehlers enthält, ist an den Pins SYO0–SYO7 für das System verfügbar. Die Fehlerlokalisierung kann durch das Speichern des Syndromwortes und der Fehleradresse vollendet werden.

Schreibzyklus

Bei einem vollständigen Schreibzyklus wird das vollständige Wort, unter Umgehung des U 80608, direkt in den RAM geschrieben. Die gleichen Daten liegen an den Eingängen WDI des U 80608, der die Checkbits generiert, an. $\overline{\text{BM0}}$ und $\overline{\text{BM1}}$ müssen Low sein,

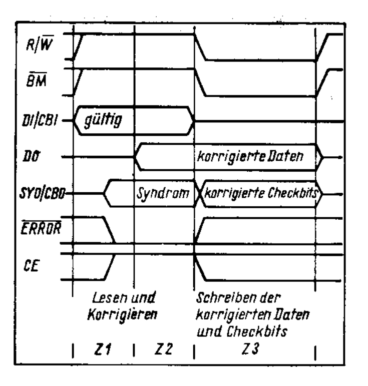


***Bild 2** Ablauf eines Read-Modify-Write-Zyklus mit Ein-Bit-Fehler in einem EDC-RAM-System mit U 80608 und U 80610*

damit die Ausgänge DO hochohmig sind. Die 5 bis 8 Checkbits erscheinen an den Pins CBO und werden im selben Schreibzyklus in den Checkbit-RAM geschrieben. In einem Multi-Chip-System berechnet der Master die Checkbits unter Verwendung der Teilparitäten der Slaves.
Beim Byte-Schreiben wird ein Teil (8 Bit) des Datenwortes überschrieben und der andere beibehalten. Das wird durch die Ausführung eines Read-Modify-Write-Zyklus ermöglicht. Das komplette alte Datenwort wird in den U 80608 gelesen und korrigiert. Dabei wird das Syndrom intern mit der H/L-Flanke von R/$\overline{W}$ gespeichert. Nur der Teil des Datenwortes (spezifiziert durch die Eingänge $\overline{\text{BM0}}$, $\overline{\text{BM1}}$), der nicht verändert wird, erscheint an den Ausgängen DO. Der Teil, der überschrieben wird, geht über den Systembus zum RAM. Der U 80608 berechnet die Checkbits des neuen Wortes unter Verwendung des intern gespeicherten Bytes vom vorausgegangenen Lesen und des neuen Bytes vom Systembus und schreibt diese anschließend in den Speicher.

Read-Modify-Write-Zyklus

Bild 2 stellt den Ablauf eines RMW-Zyklus mit Ein-Bit-Fehler in einem EDC-RAM-System mit U 80608 und U 80610 dar. Nach der Fehlererkennung kann der U 80608 dazu verwendet werden, den RAM-Controller zu veranlassen, den Read-Zyklus in einen RMW-Zyklus zur Fehlerkorrektur zu überführen. Das reduziert die Wahrscheinlichkeit des Auftretens von Multi-Bit-Fehlern bei nachfolgenden Lesezyklen. Diese Korrektur wird durch die Ausführung von RMW-Zyklen erreicht.
Der RMW-Zyklus wird durch den R/$\overline{W}$-Eingang gesteuert. Nach oder während eines Lesezyklus prüft der System-DRAM-Controller oder die CPU die Ausgänge $\overline{\text{ERROR}}$ und CE, um festzustellen, ob ein korrigierbarer Fehler vorliegt. Ist das der Fall, muß der DRAM-Controller oder die CPU den R/$\overline{W}$-Eingang des U 80608 auf Low legen. Damit wird der EDC veranlaßt, das generierte Syndrom zu speichern und die korrigierten Checkbits auf die CBO-Ausgänge zu legen. Die korrigierten Daten sind an den DO-Pins verfügbar. Der DRAM-Controller schreibt nun die korrigierten Daten und die dazugehörigen Checkbits in den Speicher. Der U 80608 kann den Read-Modify-Write-Zyklus in einem oder in zwei Speicherzyklen durchführen. Sind zwei Zyklen erforderlich, werden die U 80608-Latches zum Speichern der Daten- und der Checkbits vom Lesezyklus für den folgenden Lese- und Schreibzyklus genutzt. Der DRAM-Controller U 80610 ermöglicht, den RMW-Zyklus in *einem* Speicherzyklus durchzuführen.

Speicherinitialisierung

Ein Speichersystem, das mit einem EDC arbeitet, erfordert nach dem Einschalten eine Initialisierung des Speichers. Dadurch werden gültige Daten- und Checkbits im Speicher erzeugt. Der U 80608 unterstützt die Speicherinitialisierung durch die Write-Zero-Funktion. Beim Aktivieren den $\overline{\text{WZ}}$-Pins setzt der U 80608 im laufenden Speicherzyklus alle Datenbits auf Low und schreibt die zugehörigen Checkbits in den Speicher. Ein RAM-Controller kann durch diese Funktion nach dem Zuschalten der Betriebsspannung den RAM mit gültigen Daten- und Checkbits füllen. Grobe Speicherfehler, zum Beispiel wenn die Daten- und die Checkbits alle Low oder High sind, werden als unkorrigierbare Fehler erkannt.

Multi-Chip-Systeme

Ein einzelner U 80608 kann 8 oder 16 Datenbits und 5 oder 6 Checkbits handhaben. Fünf U 80608 können für Speicher mit einer Datenbreite von bis zu 80 Bit mit 8 Checkbits kaskadiert werden. In einem solchen Multi-Chip-System fungiert ein U 80608 als Master und die restlichen (maximal 4) als Slaves. Als Beispiel soll ein Lesezyklus in einem 32-Bit-System mit einem Master und einem Slave betrachtet werden. Der Slave berechnet die Teilparität (Partial Parity) seines Teilwortes (16 Bit) und gibt sie über die PPO-Pins an den Master weiter. Der Master verknüpft die Teilparität des Slaves mit der Parität, die er von seinem Teilwort berechnet hat, um das Syndrom zu generieren. Der Slave übernimmt das Syndrom vom Master, um die Fehlerkorrektur durchzuführen. In Systemen mit mehr als einem Slave müssen die Teilparitätsausgänge der Slaves extern durch XOR-Gatter verknüpft werden. Write- und Read-Modify-Write-Zyklen werden analog ausgeführt. Mittels verschiedener Pins wird definiert, ob ein U 80608 als Master oder als Slave fungiert.

Systemkonfiguration U 80608/ U 80610

Der U 80608 ist für die direkte Zusammenarbeit mit dem DRAM-Controller U 80610 vorbereitet. Dieser Schaltkreis erlaubt die Steuerung eines Dual-Port-Speichers. Die Kombination von U 80608 und U 80610 ermöglicht die Funktionen Fehlerkorrektur im Speicher während des Refreshs (automatic scrubbing), erweitertes RAS/CAS-Zeitverhalten für Read-Modify-Write in einfachen Speicherzyklen und automatische Speicherinitialisierung nach dem Reset. Zusammen bilden diese beiden Chips ein komplettes fehlerkorrigierendes dynamisches Dual-Port-RAM-Subsystem.

☒ KONTAKT ☏

VEB Mikroelektronik Erfurt, Forschungszentrum, Applikationszentrum Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76, App. 25

DRAM-Controller U 80610

Wilfried Schmidt
VEB Mikroelektronik „Karl Marx"
Erfurt, Forschungszentrum

Vorbetrachtungen

Der Adreßraum gegenwärtiger Mikroprozessoren (K 1810 WM86, U 80601, MC 68000) ist im Vergleich zu konventionellen Mikroprozessoren (U 880, K 580 IK80, MC 6800) enorm angewachsen. Der physisch verfügbare Speicherraum in entsprechenden Mikrorechensystemen, beispielsweise PC/AT für CAD- oder CAE-Anwendungen, erreicht häufig schon zwei Megabyte. Derart große Speicherräume sind zur Zeit ausschließlich mit dynamischen Speicherschaltkreisen kostengünstig realisierbar.

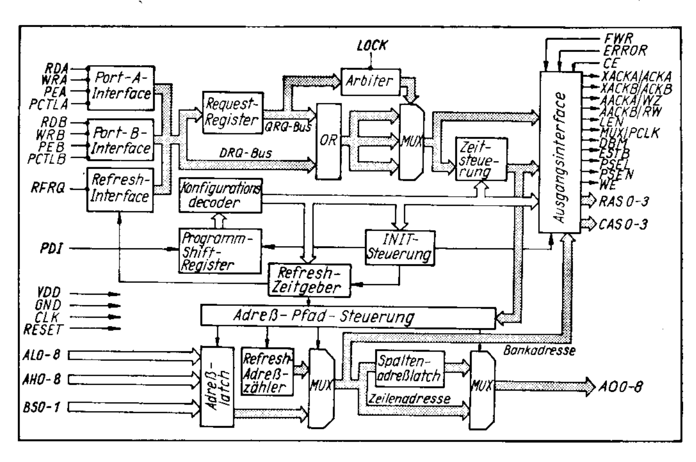


Bild 1 Blockschaltbild des U 80610

Bild 2 Pinbelegung

Beim Entwurf solcher Speichersysteme gibt es jedoch häufig Probleme, wenn einerseits hohe Anforderungen an die Leistungsfähigkeit, den Funktionsumfang und die Zuverlässigkeit gestellt werden und diese andererseits noch einfach und kompakt strukturiert ausfallen sollen. Zur Ansteuerung dynamischer Speicher ist auf Grund ihrer spezifischen Wirkungsweise ein erhöhter Hardwareaufwand nötig.

Integrierte DRAM-Controller (DRC) tragen zur Vereinfachung der Entwurfsproblematik bei, weil sie das Auffrischen der DRAMs übernehmen und alle erforderlichen Signale im Speicher- und im Mikroprozessor-Interface mit hoher zeitlicher Konstanz bereitstellen. Die Integration aller Funktionen ermöglicht beim Systementwurf entscheidende Reduzierungen bei der Platinenfläche und bei der Energieaufnahme. Gleichzeitig wird die Systemzuverlässigkeit erhöht. Der U 80610 ist ein programmierbarer DRAM-Controller für die Ansteuerung dynamischer Speicherschaltkreise. Er unterstützt den Anschluß von 16-, 64- und 256-KBit-DRAMs und kann einen Adreßraum von maximal 2 MByte verwalten. Der U 80610 besitzt zwei Ports, die hinsichtlich des Prozessorinterfaces und der Zugriffsmodalitäten konfigurierbar sind. Dadurch ist es mit vertretbarem Hardwareaufwand möglich, Dual-Port-RAM-Strukturen aufzubauen, wie sie in modernen Busarchitekturen zu finden sind. Die Kommunikation der beiden Ports mit dem jeweiligen Prozessor kann synchron oder asynchron erfolgen.

Der U 80610 erlaubt RAM-Zugriffe ohne Wartezyklen und führt bei entsprechender Programmierung eine RAM-Initialisierung durch. Weiterhin existiert in vielen Fällen die dringende Notwendigkeit, die gespeicherten Daten hardwaremäßig zu überprüfen und bei Bedarf Korrekturen vorzunehmen; dieses Datensicherungskonzept wird unterstützt. Hierfür besitzt der U 80610 neben den Port- und RAM-Interfaces noch ein ECC-Interface (Error Checking and Correction) und unterstützt damit den Anschluß des EDC-Controllers U 80608 (Error Detection and Correction). Die Einheit von RAM-Controller (U 80610) und EDC-Controller (U 80608) ermöglicht den einfachen Aufbau großer Speicherarrays und bietet darüber hinaus die Möglichkeit, korrigierbare RAM-Fehler zu beseitigen.

Aufbau und Wirkungsweise

Übersicht über die Funktionsblöcke

Der DRC besitzt verschiedene Interfaces, um mit dem Mikroprozessor und dem EDC auf der einen Seite und dem anzusteuernden DRAM auf der anderen Seite kommunizieren zu können (siehe Bild 1):

- Port A-Interface
- Port B-Interface
- Refresh-Interface
- Ausgangsinterface:
  - ECC-Interface
  - RAM-Interface
  - Quittungssignale.

Die Pinbelegung und das Schaltzeichen des U 80610 können den Bildern 2 und 3 entnommen werden.

Port-A/B-Interface

Die beiden Ports A und B des DRCs sind gleich aufgebaut und ermöglichen den Anschluß der Mikroprozessoren K 1810 WM86, U 80601 sowie kompatibler Typen. Über diese Ports wird die Kommunikation des Controllers mit den Mikroprozessoren gesteuert. Hinsichtlich des Prozessorinterfaces und der Kommunikationsprotokolle ist das Verhalten dieser Ports konfigurierbar.

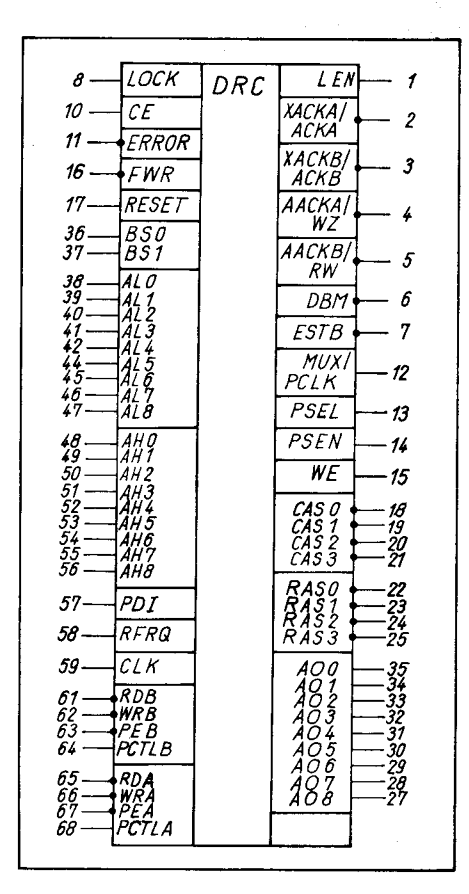
Refresh-Interface

Neben den oben erwähnten Mikroprozessor-Ports ist ebenfalls die interne Refresh-Logik berechtigt, auf den verwalteten Speicher zuzugreifen. Für den konfliktfreien Zugriff auf die gemeinsamen Speicherressourcen sorgt ein interner Arbiter. Die Konfiguration des Refresh-Interfaces erfolgt durch Abtastung des RFRQ-Pins mit der fallenden Flanke von RESET. Je nach Programmierung wird dabei der interne Refresh-Timer aktiviert oder bleibt für das Auffrischen der Speicher ungenutzt. Der als programmierbarer Teiler ausgelegte Refresh-Timer übernimmt die Steuerung der *Warm-up-Phase* und in den Betriebsarten mit intern generiertem Refresh die Anforderung der Zyklen zum Auffrischen der dynamischen Speicher. Dabei wird die Betriebsfrequenz des DRC und die programmierte Refresh-Periode berücksichtigt.

ECC-Interface

Für den Aufbau gesicherter Speicherstrukturen besitzt der DRC ein spezielles Interface und unterstützt somit den Anschluß eines entsprechenden EDC-Controllers. Der EDC-Schaltkreis überwacht den Datenverkehr

Bild 3 Schaltzeichen



zwischen Prozessor und Speicher. Im Fehlerfall wird der aktuelle Zyklus zu einem Read-Modify-Write-Zyklus verlängert; damit können korrigierbare Fehler beseitigt werden.

RAM-Interface
Das RAM-Interface ist ebenfalls Bestandteil des Ausgangsinterfaces. Herausgeführt sind hier die Adressen sowie die RAS-, CAS- und Steuersignale für den unmittelbaren Anschluß von dynamischen Speichern einschließlich der Bankselektsignale BS0 und BS1.

Betriebsarten des DRAM-Controllers
Kommando- und Statusmodus
Der logische Zustand an den Anschlüssen PCTLA und PCTLB zum Zeitpunkt der fallenden Flanke des asynchronen RESET-Impulses bestimmt, in welcher Interface-Betriebsart jeder der beiden Mikroprozessor-Ports programmiert wird. Bei High-Pegel erfolgt die Konfiguration im Statusmodus zur direkten Dekodierung der Statussignale der Prozessoren K 1810 WM86 und K 580 IK80, die in der Kodierung und im Zeitverhalten nahezu identisch sind. Eine Konfiguration im Kommandomodus erfolgt bei Low-Pegel am betreffenden PCTL-Pin. In dieser Betriebsart können direkt die Statussignale der CPU U 80601 dekodiert werden. Gleichfalls werden hier alle Multibus-Kommandos und die Speicheranforderungen eines lokalen Buscontrollers akzeptiert.
In Abhängigkeit von der Programmierung erfolgt die Synchronisation asynchroner Speicheranforderungen. Neben den direkten Anforderungssignalen im DRQ-Bus (direct request), die zur Zyklusauslösung bei bereits angewähltem Port verwendet werden, speichert ein spezielles Register die Anforderungen, falls der Zyklus nicht sofort ausgeführt werden kann. Die Signale des QRQ-Busses (queued request) bilden so eine Warteschlange für abzuarbeitende Speicherzyklen. Diese werden erst durch einen Zyklusstart auf dem zugeordneten Port oder im Kommandomodus durch ein INHIBIT gelöscht.

Refresh-Betriebsarten
Prinzipiell werden fünf Refresmodi vom vorgestellten DRC unterstützt:
- intern generierter Refresh
- intern generierter Refresh mit Ausfallschutz
- extern generierter Refresh ohne Ausfallschutz
- Burstrefresh
- kein Refresh.

Durch die Abtastung des Eingangs RFRQ mit der fallenden RESET-Flanke erfolgt die Programmierung des Refreshmodus. Bei High-Pegel wird der Refresh-Timer für eine interne Generierung von Refresh-Anforderungen aktiviert. In diesem Fall ist eine externe Anforderung von Refresh-Zyklen bei gleichzeitiger Ausfallüberwachung und eine ausschließlich interne Refresh-Steuerung durch den U 80610 möglich. Bei Low-Pegel am RFRQ-Pin bleibt der Refresh-Timer inaktiv. Der Anwender hat dadurch die Wahl zwischen der Generierung von einzelnen Refresh-Zyklen ohne Ausfallschutz und Burstrefresh zum Auffrischen der dynamischen Speicher.

NEC- und ECC-Modus
Der logische Zustand am PDI-Pin beim Rücksetzen des DRC bestimmt, ob der DRC in einer Konfiguration mit oder ohne Fehlerüberwachung und -korrektur eingesetzt wird. Da einige Anschlüsse zur optimalen Anpassung an das Gesamtsystem andere Funktionen übernehmen und deshalb die internen Schaltungen zur Ablaufsteuerung bereits vor Abschluß der Programmierphase konfiguriert werden, nimmt das ECC-Programm-Bit im Programm-Wort des U 80610 eine Sonderstellung ein. Bei Low-Pegel am Anschluß PDI erfolgt die Programmierung im Nicht-ECC-Modus (NEC), bei High-Pegel im ECC-Modus.

Betriebsarten zum Test
Zur Vereinfachung der Funktionskontrolle bietet der U 80610 zwei Testmöglichkeiten. Diese sind vorrangig für den Hersteller vorgesehen und werden an dieser Stelle nicht näher erläutert.

Funktionsbeschreibung
Anschluß der Mikroprozessoren
Der U 80610 ist für eine direkte Ankopplung an die Mikroprozessorsysteme K 1810 WM86 und U 80601 vorgesehen. Die Statussignale dieser Prozessoren können im Interesse einer maximalen Geschwindigkeit direkt dekodiert und auch unabhängig vom Systemtakt verarbeitet werden. Aber auch an andere Busstrukturen mit getrennten Strobes für Lese- und Schreibanforderungen kann der U 80610 leicht angepaßt werden. Heute werden vielfach Multiprozessorsysteme eingesetzt, die mit einem anteilig genutzten Speicherbereich arbeiten, um eine höhere Verarbeitungsgeschwindigkeit und eine höhere Zuverlässigkeit zu erreichen. Durch die Dual-Port-Architektur des U 80610 wird ein einfacher Aufbau solcher Systeme unterstützt.

Synchroner und asynchroner Betrieb
Die Begriffe *synchron* bzw. *asynchron* charakterisieren das Eingangsverhalten des DRAM-Controllers. Die Kommandos können dabei in bezug auf seinen Arbeitstakt eine genau definierte Phasenlage haben. Solche synchronen Signale sind direkt im Interface einsetzbar. Besteht jedoch keine festgelegte Beziehung zum Takt, zum Beispiel bei einem Speicherzugriff außerhalb der lokalen Prozessorumgebung, dann müssen alle eintreffenden Kommandos zunächst synchronisiert werden. Danach ist die weitere Behandlung aller Anforderungen identisch. Die On-chip-Synchronisation wird durch zwei Flip-Flop-Stufen realisiert, die mit der gleichen Taktflanke getriggert werden. Die Synchronisationszeiten liegen aus diesem Grund zwischen minimal einer und maximal zwei Taktperioden. Wegen der dann durchschnittlich um etwa 25% höheren Zugriffszeit sollte synchronen Stufen, wann immer das möglich ist, der Vorrang gegeben werden.

Anforderungs- und Steuersignale
Zur Auslösung eines Speicherzyklus verfügt der U 80610 je Port über vier Eingänge. Die Eingangssignale RD, WR und PCTL sind mit den äquivalenten Statussignalen des unterstützten Mikroprozessors, mit den Kommandoleitungen eines Buscontrollers oder mit den Multibus-Signalen zu verbinden. Im Fast Cycle Mode (U 80 601-Interface) sollte PCTL fest auf Masse gelegt werden. Im Kommandomodus kann über dieses Signal zusätzlich in den Zyklusablauf eingegriffen werden. PCTL wird dazu abgefragt und gibt einen angeforderten Zyklus frei oder verbietet diesen. Erfolgt ein INHIBIT (ein nachträgliches Sperren des Speichers) bereits vor dem Zyklusstart, dann wird die entsprechende Anforderung aus der Warteschlange gelöscht. Ein bereits begonnener Speicherzyklus wird dagegen immer abgeschlossen. Alle Lese- und Schreibanforderungen an den Speicher müssen durch die Signale PEA bzw. PEB bestätigt werden. Da diese Freigabesignale in der Regel aus der Adresse dekodiert werden, ist eine unkomplizierte Staffelung mehrerer DRAM-Controller zur Ansteuerung eines größeren Speicherbereichs möglich.
Bei einer Daueraktivierung der Eingänge RD/WR werden keine aufeinanderfolgenden Zyklen generiert. Bevor eine weitere Speicheranforderung (über das Port) anerkannt wird, müssen die Kommandos für mindestens eine Taktperiode inaktiv werden. Durch daueraktive Kommando- oder Statussignale kann auch der Zugriff auf den Speicher vom jeweils anderen Port aus nicht verhindert werden.

K 1810 WM86-Interface
Bei der Verwendung des K 1810 WM86 arbeitet der U 80610 im *Slow Cycle Mode*. Für einen Betrieb ohne Wartezustände ist der U 80610 direkt an die Statusleitungen des Prozessors anzuschließen und synchron zu takten. Weiterhin muß das Quittungssignal EAACK mit dem SRDY-Eingang des Prozessors verbunden werden. Der Anschluß der Adreßeingänge des U 80610 an den Daten-/Adreßbus des Mikroprozessors erfolgt über geeignete Latches und Demultiplexer (z. B. DS 8283).

U 80601-Interface
Zusammen mit dem U 80601 arbeitet der U 80610 im *Fast Cycle Mode*. Für einen Betrieb ohne Wartezustände muß der U 80610 direkt an die Statusleitungen des Prozessor angeschlossen und synchron getaktet werden. Das Quittungssignal EAACK ist mit dem SRDY-Eingang eines DS 82284 (Clock Generator and Ready Interface) zu verbinden. In Single-Port-Anwendungen können die Adreßeingänge direkt an den Adreßbus des Prozessors angeschlossen werden, wenn am Port A angekoppelt wird.

Dual-Port-Betrieb
Das Dual-Port-Konzept des DRC U 80610 ermöglicht zwei unabhängigen Prozessoren den Zugriff auf einen gemeinsam genutzten Speicher. Der Controller steuert dabei den Adreß- und Datenpfad zwischen beiden Ports über spezielle Signale und realisiert durch den internen Arbiter einen konfliktfreien Zugriff. Damit beide Ports synchron betrieben werden können, müssen die Arbeitsfrequenzen der Mikroprozessoren phasensynchronisiert sein. Sind diese Voraussetzungen nicht erfüllbar, dann muß ein Port asynchron arbeiten.

Zyklusbestätigung
In Systemkonfigurationen ohne Fehlerkorrektur liefert der U 80610 zwei Quittungssignale für jedes Port, AACK und XACK. In Abhängigkeit von der Programmierung des DRC werden die Quittungssignale für den jeweiligen Betrieb optimiert ausgegeben.
In ECC-Systemen ist über ACKA bzw. ACKB immer nur ein Bestätigungssignal je Port verfügbar. Ob dieses als AACK oder XACK ge-

Fortsetzung auf Seite 147

Einführung in Forth-83

Teil 2

Dr. Hartmut Pfüller (Leiter),
Dr. Wolfgang Drewelow, Dr. Bernhard Lampe,
Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

Übrigens ...

... sollte die im Jahre 1969 von Charles Moore an einem Observatorium in Charlottsville entwickelte Programmiersprache Fourth heißen und zum Ausdruck bringen, daß sie für die 4. Computergeneration konzipiert wurde. 6stellige Bezeichnungen waren aber auf der von Moore genutzten Anlage IBM 1130 noch nicht möglich, so daß er kurzerhand einen Buchstaben wegließ. Damit entstand Forth als Name für eine Echtzeit-Programmiersprache mit einer bis zu zehnmal höheren Geschwindigkeit gegenüber Basic. *MP*

2. Kompilieren in Forth

Herkömmliche Compiler sind in sich abgeschlossene Rechenprogramme, die Quelltext so übersetzen, daß das Übersetzungsprodukt (oder auch Kompilat) schließlich in den Speicher gebracht und dort abgearbeitet werden kann. In Forth ist der Übersetzungsvorgang nicht derart in einem separaten Compiler zentralisiert. Hier entsteht das Kompilat durch leicht nachvollziehbare Aktionen des Textinterpreters, wobei sich die zu übersetzenden Worte selbst noch aktiv am Übersetzungsvorgang beteiligen können.
In diesem Teil werden zunächst die Befehle für den Zugriff zum Rechnerspeicher zusammengestellt. Darauf aufbauend folgen Ausführungen zum Übersetzungsvorgang als Wörterbucherweiterung, über Worte zur Programmstrukturierung und zum Umgang mit Teilwörterbüchern.

2.1. Speicherzugriff

2.1.1. Adressen

Für den Übersetzungsprozeß und die Ablage des entstandenen Kompilats sind Zugriffsmöglichkeiten zum Speicher erforderlich. Hierfür benutzt das Forthsystem genau dieselben Worte, die auch für den Programmierer da sind, wenn er auf Speicheradressen zugreifen möchte. Diese Worte erwarten die Speicheradresse, auf die zuzugreifen ist, als 16-Bit-Wert auf dem Parameterstapel. Damit wird die Adressierung von 64 kByte direkt unterstützt.

2.1.2. Übertragung zwischen Speicher und Stapel

Die drei Worte
C@ @ 2@
lesen Werte vom Speicher. Sie erwarten auf dem Stapel eine Adresse, entfernen diese und holen von dem damit adressierten Speicherplatz ein, zwei bzw. vier Byte zum Stapel. Auch beim Befehl C@, der nur ein Byte holt, wird als Ergebnisparameter auf dem Stapel ein 16-Bit-Wert hinterlassen; dabei sind die höherwertigen acht Bit auf Null gesetzt.
Dagegen schreiben die drei Worte
C! ! 2!
auf die im TOS (Top Of Stack) liegende Adresse den darunterliegenden Wert, und zwar auch wieder ein, zwei bzw. vier Byte. Das Wort +! erwartet im TOS ebenfalls eine Speicheradresse. Der darunter liegende Wert wird zum auf der anzusprechenden Speicherzelle liegenden 16-Bit-Wert addiert.

2.1.3. Behandlung von Speicherbereichen

Mit den Worten
CMOVE CMOVE>
kann der Inhalt von Speicherbereichen verschoben werden. CMOVE kopiert dabei zuerst das erste Byte, CMOVE> kopiert zuerst das letzte Byte des Bereichs. CMOVE> eignet sich also zum Aufwärtsverschieben von sich überlappenden Speicherbereichen, was auch durch das Zeichen > am Namen angedeutet werden soll. Das Wort FILL füllt einen Speicherbereich mit dem Byte, das als Parameter im TOS liegt. Größe und Anfangsadresse des Bereichs liegen darunter.

2.2. Die Erweiterung des Wörterbuchs

Befehlsworte werden immer dann nicht ausgeführt, sondern übersetzt, wenn der Kompiliermodus eingeschaltet ist, also zum Beispiel zwischen Doppelpunkt und Semikolon. In diesem Zustand wird das Forthsystem gewissermaßen davon unterrichtet, welche Befehlsabfolge zu einem neu definierten Wort gehören soll. Der Mechanismus, nach dem diese Informationen übersetzt werden, ist denkbar einfach: Nachdem der neue Name im Klartext als nächster Eintrag an das bisherige Ende des Wörterbuchs dazugeschrieben wurde, werden dann der Reihe nach die Codeanfangsadressen der als Programminhalt notierten Worte angehängt. Auf diese Weise wird das Wörterbuch um einen neuen Eintrag erweitert; es wächst.

2.2.1. Wortnamen

Definitionsworte tragen bei ihrer Ausführung einen zusätzlichen Namen in das Wörterbuch ein. Die Form dieser Eintragung im Wörterbuch unterscheidet sich nicht von den Eintragungen, die vorher schon zum Forthsystem gehörten.
Wegen dieser Einheitlichkeit kann das gesamte Forthsystem als ein ausschließlich mit den eigenen Mitteln definiertes Programm angesehen werden. Professionelle Forthsysteme werden auch genau auf diese Weise erzeugt: Sie sind vollständig in Forthquellcode definiert und entstehen als Ergebnis eines Übersetzungsvorgangs mittels Forth. Für die im Standard Forth-83 definierten Funktionen sind möglichst sinnfällige Namen festgelegt. Für den Programmierer gibt es bei der Wahl der Namen für seine eigenen Worte nur die Einschränkung, daß der Name nicht mehr als 31 Zeichen enthalten darf. Sowohl im Ausführungs- als auch im Kompiliermodus ist der nur wenige Zeilen lange Textinterpreter dasjenige Systemprogramm, das die einzelnen Worte aus dem Eingabestrom entgegennimmt und je nach Betriebsmodus aufruft (d. h. ausführt) oder kompiliert. Von den Worten im Quellcode nimmt der Textinterpreter im allgemeinen an, daß sie schon als Einträge im Wörterbuch vorhanden sind. Allerdings sind an dieser Stelle zwei Ausnahmen zu erwähnen: Als erste Ausnahme sind im Eingabestrom Zahlen zulässig, da der Textinterpreter bei unbekannten Worten zunächst versucht, diese als numerische Werte aufzufassen und zu konvertieren. Als zweite Ausnahme kann es geschehen, daß ein vom Textinterpreter aufgerufenes Wort die weitere Analyse des Eingabestroms zeitweilig selbst *in die Hand nimmt*. Dieses Wort kann dann nachfolgende Worte im Quelltext als Parameter werten und eventuell gar nicht erst im Wörterbuch suchen. So arbeiten beispielsweise Definitionsworte, wenn sie für den nachfolgenden Namen einen neuen Wörterbucheintrag einrichten.

2.2.2. Definition von Datenworten

a) Unveränderliche Zahlenwerte
Ein symbolischer Name für einen unveränderlichen Zahlenwert kann mit dem Definitionswort CONSTANT vereinbart werden. CONSTANT erledigt bei seiner Aktivierung zwei Aufgaben: Es schreibt für den neu definierten Wortnamen einen neuen Eintrag ins Wörterbuch und holt den obersten Wert vom Stapel, um ihn im Wörterbuch bei dem neuen Namen abzulegen:

```
26  CONSTANT  BUCHSTABEN(cr) ok
10  CONSTANT  ZIFFERN(cr) ok
BUCHSTABEN ZIFFERN + CONSTANT ZEICHEN(cr) ok
BUCHSTABEN  .(cr) 26 ok
ZIFFERN  .(cr) 10 ok
ZEICHEN  .(cr) 36 ok
```

Der unterstrichene Teil kennzeichnet die Eingabe der Entertaste (cr) und die Rechnerreaktion bis (ok). Nachdem ein Konstantenname definiert wurde, ist die Benutzung dieses Namens funktionell gleichwertig mit der direkten Benutzung der betreffenden Zahl. Hier soll außerdem deutlich werden, daß es nicht etwa syntaktische Vorschrift ist, vor dem Wort CONSTANT eine Zahl zu notieren. Der Programmierer muß nur wissen, daß CONSTANT den augenblicklich auf dem Parameterstapel befindlichen Wert holt und diesen als vereinbarten Konstantenwert ansieht. Wie der Programmierer diesen Wert gewinnt, gehört zur Vorgeschichte und ist für die Funktion von CONSTANT vollständig gleichgültig. Es ist also möglich, den zu vereinbarenden Konstantenwert erst durch ein (beliebig komplexes) Programm ermitteln zu lassen. Weiterhin ist es nützlich zu wissen, daß ein definiertes Datenwort wie zum Beispiel ZIFFERN selbst ein aktives Wort ist. Zu verstehen ist das so, daß das Wort ZIFFERN bei seiner Ausführung völlig selbständig die weitere Programmabarbeitung übernimmt, in eigener Zuständigkeit den Wert 10 zum Para-

meterstapel schafft und anschließend die weitere Programmabarbeitung wieder an das Forthsystem zurückgibt.

b) Veränderliche Zahlenwerte
Symbolische Namen für veränderliche Zahlenwerte werden mit dem Definitionswort VARIABLE vereinbart.

VARIABLE TEILE(cr) ok

Das Wort VARIABLE hat jetzt ans Wörterbuch einen Eintrag mit dem Namen TEILE und dem Code für veränderliche Zahlenwerte angehängt. Dabei wurde eine Speicherzelle freigehalten, in der der jeweils aktuelle Wert von TEILE eingetragen ist. Wenn das Wort TEILE ausgeführt wird, wird automatisch der Codeteil für veränderliche Zahlenwerte aktiviert, und der seinerseits schafft genau die Adresse der freigehaltenen Speicherzelle zum Parameterstapel. Der Programmierer benutzt nun diese Adresse, um mit den gewöhnlichen Worten für den Zugriff auf Speicheradressen den Variablenwert zu setzen oder zu lesen:

```
100  TEILE  !(cr) ok
50  TEILE  +!(cr) ok
TEILE  @  .(cr) 150 ok
-70  TEILE  +!(cr) ok
TEILE  @  .(cr) 80 ok
```

Der Name TEILE ist damit nichts anderes als eine menschliche Merkhilfe für die zugeteilte Speicheradresse. Falls der Programmierer sich dafür interessiert, welche konkrete Speicherzelle das Forthsystem für die Variable TEILE reserviert hat, kann er sich diese Speicheradresse anzeigen lassen.

TEILE .(cr) 32373 ok

Der Variablenname hinterläßt ja bei seiner Ausführung als eigene Aktion die zugeteilte Adresse auf dem Stapel, und der Punkt gibt diesen Wert aus. Ob der Programmierer in dieser angenommenen Situation nun mittels des Namens TEILE oder mittels der Zahl 32373 auf die bewußte Speicherzelle zugreift, ist funktionell gleichwertig. In Quelltexten empfiehlt sich natürlich der Gebrauch des Namens, um in jedem Fall die richtige Bezugnahme zu sichern. Die Verfügbarkeit der Variablenadresse ermöglicht ein flexibles Reagieren auf wechselnde Anforderungen. Angenommen, es gäbe ein bewährtes größeres Programm zur Bestandsverwaltung, mit dem nach dem obigen Muster Zugänge und Abgänge mittels der Variablen TEILE registriert werden. Nun möge sich die Notwendigkeit ergeben, nicht nur wie bisher eine einzige, sondern zwei verschiedene Sorten dieser Teile zu verwalten, beispielsweise solche aus Holz und solche aus Plast. Sicherlich wird man dafür zwei verschiedene Zählvariablen (z. B. #HOLZ und #PLAST) vorsehen müssen. Trotzdem kann bei entsprechend zweckmäßigem Vorgehen gesichert werden, daß der bewährte Programmbestand weitgehend unverändert bleibt. Die Methode besteht darin, das Verhalten von TEILE nachzubilden und bei Bedarf nur die Sorte zu wechseln. Dazu wird eine zusätzliche Variable MATERIAL definiert, in der die jeweils aktuelle Sorte gespeichert ist. Mit den Befehlen HOLZ und PLAST wird dann bei Bedarf die Sorte gewechselt, während die gesamte Zählung weiter mit dem bisherigen Programm für TEILE erledigt werden kann (siehe Bild, vgl. auch Abschnitt 2.4.3.).

```
VARIABLE  #HOLZ        0  #HOLZ   !(cr) ok
VARIABLE  #PLAST       0  #PLAST  !(cr) ok
VARIABLE  MATERIAL      #HOLZ MATERIAL !(cr) ok
: HOLZ    ( ===> )  #HOLZ  MATERIAL ! ;(cr) ok
: PLAST   ( ===> )  #PLAST MATERIAL ! ;(cr) ok
:  TEILE    ( ===> a)  MATERIAL  @  ;(cr) ok
```

Die Variablen haben nach ihrer Definition einen unbestimmten Inhalt und werden deshalb geeignet initialisiert. TEILE ist jetzt keine Variable mehr, sondern ein Programm, das bei Aufruf eine der Variablenadressen #HOLZ oder #PLAST liefert. Wenn das Verwaltungsprogramm aber nur voraussetzt, daß der Aufruf von TEILE die Adresse der Zählvariablen zum Stapel liefert, dann muß es weiter korrekt arbeiten. Für den Benutzer des Programms hat sich nur geändert, daß er jetzt bei Bedarf eine bestimmte Sorte anwählen kann:

```
PLAST  12  TEILE  +!  20  TEILE   +!(cr) ok
HOLZ   65  TEILE  +!  -15  TEILE  +!(cr) ok
PLAST  TEILE  @  .(cr) 32 ok
HOLZ   TEILE  @  .(cr) 50 ok
```

2.2.3. Definition von Programmworten

Die übliche Art und Weise, Forthprogramme zu schreiben, besteht in der Definition von Programmworten mittels des Doppelpunkts. Gute Programmierer finden bei der Problemanalyse heraus, in welche kleinsten funktionellen Einheiten das Problem zerlegt werden kann. Für Forthsoftware ist charakteristisch, daß solche funktionellen Einheiten als sehr kleine eigenständige Programme definiert sind. Ist in einem Programm für elektrische Schaltungen beispielsweise die Parallelschaltung von Widerständen zu berechnen, böte sich an, ein Programm für einen Parallelschaltungsoperator vorzusehen:

: || (**r1 r2** ===> **r1||r2)(cr) ok**
OVER OVER + ∗/ **;(cr) ok**

Gesamtwiderstand dreier paralleler Widerstände im Wert von ein, zwei und drei kOhm kann damit folgendermaßen ermittelt werden.

1000 2000 3000 || || .(cr) 545 ok

Der Doppelpunkt als Definitionswort hat den neuen Namen || ins Wörterbuch eingetragen. Nach der Kompilierung der vier Worte, die das Programm ausmachen, schaltet das Semikolon wieder den Ausführungsmodus ein. Mit dem Definieren von Programmen wie || für die elementaren Bestandteile des Problems entsteht nach und nach ein Vorrat an Worten, der seinerseits wieder benutzt werden kann, um die nächsthöheren Niveaus der Aufgabe problemnah zu beschreiben.

2.2.4. Streichen von Definitionen

Im Laufe einer Forthsitzung kann die Lage eintreten, daß das Wörterbuch mit Probierversionen beladen ist, die inzwischen nicht mehr gebraucht werden. Dann gibt es die Möglichkeit, mittels des Wortes FORGET von einem anzugebenden Namen an das gesamte Ende des Wörterbuchs wieder „abzuschneiden". In der folgenden Zeile wird der Eintrag MATERIAL zusammen mit allen nachfolgenden Einträgen aus dem Wörterbuch getilgt.

FORGET MATERIAL(cr) ok

2.3. Programmstrukturierung

Die bisherigen Beispiele enthielten nur reine *Geradeausprogramme*. So trivial sind reale Programme in aller Regel nicht, vielmehr werden Möglichkeiten für Entscheidungen, Verzweigungen und Wiederholungen benötigt.

2.3.1. Logische Werte

Zahlen auf dem Parameterstapel können auch als logische Werte angesehen werden. Dabei gilt in Forth die Vereinbarung, daß die Zahl Null den logischen Wert *falsch* bedeutet. Jede Zahl ungleich Null gilt dagegen als logischer Wert *wahr*. Logische Werte werden im Laufe der Programmabarbeitung zum Beispiel als Ergebnisse von Vergleichsoperationen geliefert. Ein Vergleichsoperator arbeitet dabei nach demselben Schema, wie die arithmetischen Operatoren: Die beiden zu vergleichenden Zahlen liegen oben auf dem Stapel. Der Vergleichsoperator entfernt die Zahlen und hinterlegt auf dem Parameterstapel denjenigen logischen Wert, den er als Ergebnis des Vergleichs herausgefunden hat. Wenn sich der logische Wert *wahr* ergibt, dann erzeugen die Operatoren dafür eine *−1* (alle Bits auf *1* gesetzt). Als Operatorname für den Test auf Gleichheit wird naheliegenderweise das Gleichheitszeichen = benutzt. Ähnlich selbsterklärend wirken die Operatoren für *größer als* und *kleiner als*, wie das Bild zeigt.

```
1   2   =   .(cr) 0 ok
3   3   =   .(cr) -1 ok
-5  0   <   .(cr) -1 ok
-5  0   >   .(cr) 0 ok
```

Häufig tritt der Fall auf, daß eine Zahl mit Null verglichen werden soll. Vom Standard sind für diese Fälle drei Operatoren (**0**= **0**> und **0**<) vorgesehen. Alle bisher genannten Vergleichsoperatoren sehen die zu vergleichenden Zahlen als vorzeichenbehaftete 16-Bit-Zahlen an. Wenn die zu vergleichenden Zahlen als vorzeichenlos angesehen werden sollen, muß der Operator U< benutzt werden.

30000 40000 U< .(cr) −1 ok

Für den Vergleich von doppeltgenauen Zahlen sieht der Standard die folgenden vier Operatoren vor:

D0= D= D< DU<

2.3.2. Bitoperationen

Zur bitweisen Verknüpfung von zwei logischen Eingangswerten zu einem logischen Ergebniswert dienen die Worte

AND OR XOR

Das Wort NOT bildet die bitweise Negation (Einerkomplement) eines Eingangswertes.

2.3.3. Entscheidungsstrukturen

Logische Werte (z. B. aus Ergebnissen von Vergleichen) werden häufig zur Entscheidung über Programmverzweigungen benutzt. In Forthprogrammen wird die Verzweigung mittels der Worte IF ELSE und THEN organisiert:

.... IF .... ELSE .... THEN ....

Die Syntax unterscheidet sich völlig von der

in herkömmlichen Programmiersprachen, ist aber in sich wieder sehr logisch: Bereits *vor* dem IF muß der Programmierer **alle** erforderlichen logischen Operationen erledigt haben, also zum Beispiel Vergleiche, logische Verknüpfungen ihrer Ergebnisse usw. Das Ergebnis liegt nun als logischer Wert im Sinne einer Steuerflagge (englisch: flag) auf dem Parameterstapel. Das Wort IF holt diesen logischen Wert vom Stapel, wertet ihn aus, und organisiert, wo der Programmlauf weiterzugehen hat. In diesem Sinne kann IF als Verzweigungsoperator angesehen werden, der entsprechend der Postfixnotation hinter (!) den als Steuerflagge dienenden Operanden steht. IF selbst leitet dann schon den Ja-Zweig ein und ist deshalb in der deutschen Entsprechung als **falls-ja** zu verstehen. Der Nein-Zweig beginnt mit ELSE, das in der deutschen Entsprechung als **andernfalls:** zu verstehen ist. Beendet wird die Verzweigung mit dem Wort THEN im Sinne des deutschen **danach:** Mit diesen Mitteln der Verzweigung kann beispielsweise ein Wort L. für die **freundlichere** Anzeige von logischen Werten programmiert werden:

```
:  L.          ( flag ===> )(cr)_ok
   IF      ." wahr "           ( falls ja)(cr)_ok
   ELSE    ." falsch "     ( andernfalls)(cr)_ok
   THEN  ;(cr)_ok
36  ZEICHEN  =  L.(cr)_wahr_ok
TEILE  @  0=  L.(cr)_falsch_ok
```

Der ELSE-Teil kann auch weggelassen werden:

```
:  KONTROLLE     ( ===> )(cr)_ok
   TEILE  @  0<(cr)_ok
     IF      ."  Achtung, Defizit! "(cr)_ok
     THEN  ;(cr)_ok
-90  TEILE  +!(cr)_ok
KONTROLLE(cr)__Achtung,_Defizit!_ok
```

2.3.4. Schleifen mit unbestimmter Durchlaufzahl

Wenn eine Programmpassage mehrmals abgearbeitet werden soll, kann sie zwischen die Worte BEGIN und UNTIL geschrieben werden:

.... BEGIN .... UNTIL ....

Bei der Abarbeitung des Programms konsumiert UNTIL in jedem Durchlauf einen Wert vom Parameterstapel und lenkt den Programmfluß immer dann um (zu BEGIN zurück), wenn der geholte Wert gleich Null (also *falsch*) war. Wenn UNTIL einen Wert ungleich Null (also *wahr*) auf dem Stapel vorfindet, wird der Programmfluß nicht umgelenkt, sondern hinter UNTIL fortgesetzt; damit wird die Schleife verlassen. Den Wert, den UNTIL vom Stapel abholt, muß der Programmierer durch geeignete Gestaltung des Programms bereitstellen. Im folgenden Beispiel wird von einer vorzugebenden Speicheradresse an so lange Byte für Byte ausgelesen und angezeigt, bis ein Speicherplatz mit dem Inhalt Null gefunden wird.

```
:  INHALT       ( addr ===> )
     BEGIN      ( addr                 )
        DUP     ( addr    addr         )
        1+      ( addr    addr+1       )
        SWAP    ( addr+1 addr          )
        C@      ( addr+1 byte          )
        DUP     ( addr+1 byte    byte)
        .       ( addr+1 byte          )
        0=      ( addr+1 flag          )
     UNTIL      ( addr+1               )
   DROP  ;      (                      )
```

Der hier noch einmal besonders ausführliche Stapelkommentar wird in späteren Beispielen weggelassen, sollte aber vom Leser im Zweifelsfall nachgeholt werden. Zuweilen wird eine Schleife gebraucht, wo über die Wiederholung nicht erst am Ende, sondern schon früher entschieden wird. Dafür kann die Konstruktion

... BEGIN ... WHILE ... REPEAT ...

benutzt werden. Von REPEAT wird der Programmfluß unbedingt zurück zu BEGIN gelenkt. WHILE ist dasjenige Wort, das vom Parameterstapel eine Steuerflagge holt und auswertet. Bei einem Wert ungleich Null (*wahr*) wird der Programmfluß nicht umgelenkt, sondern hinter WHILE fortgesetzt; durch REPEAT wird dann die Schleife wiederholt. Bei einem Wert gleich Null (*falsch*) wird durch WHILE der Programmfluß zu der Stelle hinter REPEAT umgelenkt; die Schleife wird also verlassen. Mit dieser Art Schleife kann das obige Programm so modifiziert werden, daß die abschließende Null nicht mit ausgegeben wird:

```
:  INHALT1      ( addr ===> )(cr)_ok
       BEGIN    DUP  C@  DUP(cr)_ok
       WHILE    .   1+(cr)_ok
       REPEAT(cr)_ok
   2DROP  ;(cr)_ok
```

2.3.5. Zählschleifen

Zählschleifen verwalten einen sogenannten *Laufindex*, für den zwei Grenzwerte vorgegeben werden. Nach jedem Schleifenzyklus wird zum Laufindex ein gewisser Wert, das *Inkrement*, addiert. Mit dem Inkrement eins arbeitet die Zählschleife

... DO .... LOOP ....

Das Wort DO erwartet auf dem Stapel den Startwert des Laufindex, und darunter dasjenige Limit, für das die Schleife *nicht* mehr abgearbeitet werden soll. LOOP erhöht nach jedem Zyklus den Laufindex um eins und prüft, ob das Limit erreicht ist. Wenn das Limit noch nicht erreicht ist, wird die Abarbeitung hinter DO wiederholt; ansonsten wird die Schleife verlassen und das Programm hinter LOOP fortgesetzt. Das folgende Bild zeigt

```
:  STERNE       ( n ===> )(cr)_ok
   0  DO     ." *"(cr)_ok
      LOOP  ;(cr)_ok
5  STERNE(cr)_*****ok
```

das Benutzungsprinzip. Das DO findet oben auf dem Stapel als Anfangswert für den Laufindex eine Null und darunter das Limit. Daß diese Werte aus verschiedenen Quellen stammen, nämlich das Limit vom Dialog des Programmierens und die Null vom Programm STERNE, gehört für DO zur Vorgeschichte und ist damit nicht interessant. Wichtig ist nur das Vorhandensein der beiden Werte auf dem Stapel beim Arbeitsbeginn von DO. Wenn der Programmierer innerhalb der Schleife den aktuellen Wert des Laufindex benötigt, kann das Wort I benutzt werden, das diesen Wert auf dem Stapel hinterlegt. Mit diesen Mitteln kann ein kleines DUMP-Programm geschrieben werden:

```
:  WDUMP      ( anfang ende ===> )(cr)_ok
   SWAP  DO   I  @  .  2(cr)_ok
         +LOOP  ;(cr)_ok
```

Andere Inkremente als Eins (auch negative!) sind verwendbar, wenn die Schleife mit +LOOP abgeschlossen wird. + LOOP holt in jedem Durchlauf das aktuelle Inkrement als Zahlenwert vom Stapel und addiert es zum Index. Die Schleife wird nicht mehr wiederholt, wenn der Index die Grenze zwischen Limit und Limit-1 überschritten hat. Im folgenden Bild wird +LOOP in einem DUMP-Programm für 16-Bit-Zellen benutzt.

```
:  DUMP       ( anfang ende ===> )(cr)_ok
   SWAP  DO    I  C@  .(cr)_ok
         LOOP  ;(cr)_ok
```

Entsprechend den Regeln der strukturierten Programmierung sind alle angeführten Programmstrukturen schachtelbar. Für ineinander geschachtelte Zählschleifen kann es wünschenswert sein, innerhalb der inneren Schleife den aktuellen Wert des Laufindex der nächstäußeren Schleife festzustellen. Der Standard sieht dafür das Wort J vor. Schließlich gibt es noch die Möglichkeit, Zählschleifen vorzeitig zu verlassen. Zu diesem Zweck wird das Wort LEAVE benutzt. Wenn LEAVE erreicht wird, verzweigt der Programmfluß direkt zu der Stelle hinter LOOP (bzw. +LOOP). Um das zu demonstrieren, wird das obige DUMP-Programm so geändert, daß es vorzeitig endet, falls es auf ein Byte gleich Null trifft:

```
:  DUMP1      ( anfang ende ===> )(cr)_ok
   SWAP  DO  I  C@  DUP  .  0=(cr)_ok
                IF   DROP  LEAVE(cr)_ok
                THEN(cr)_ok
         LOOP  ;(cr)_ok
```

2.4. Die Gliederung des Wörterbuchs

2.4.1. Vokabulare

Forth bietet die Möglichkeit, inhaltlich zusammengehörige Worte in spezialisierten, eigenständigen Teilwörterbüchern, den sogenannten Vokabularen, zusammenzufassen. Jedes Vokabular erhält dabei einen eigenen Namen. Mit dem Definitionswort VOCABULARY kann der Programmierer Namen für neue Vokabulare vereinbaren. Alle vom Standard vorgesehenen Forthworte befinden sich in einem Vokabular mit dem Namen FORTH.

2.4.2. Die Suchordnung

Das Vorhandensein mehrerer Vokabulare macht Festlegungen darüber erforderlich, welche Vokabulare in welcher Reihenfolge durchsucht werden sollen. Dies wird vom Standard Forth-83 gegenwärtig noch nicht verbindlich geregelt. Nachfolgend wird ein inzwischen verbreitetes Verfahren beschrieben, das im Standard als Vorschlag enthalten ist. Um den Suchraum festzulegen, werden ein oder mehrere Vokabulare als *resident* und genau ein einziges als *transient* erklärt. Die Namen aller Worte im transienten Vokabular kann man sich anzeigen lassen, indem man WORDS aufruft. Die Suche beginnt grundsätzlich im transienten Vokabular und wird danach in den residenten Vokabularen fortgesetzt. Das Aufrufen eines Vokabularnamens führt dazu, daß es zum transienten Vokabular und damit zum ersten in der

Suchreihenfolge wird. Das Installieren einer gewünschten Suchordnung wird meist mit ONLY FORTH eingeleitet. ONLY entfernt sämtliche bisherigen Vokabulare aus dem Suchraum und erklärt ein spezielles Minimalvokabular als resident und gleichzeitig als transient. Durch das Nennen des Vokabulars FORTH wird es zum neuen transienten Vokabular. ALSO kopiert das transiente Vokabular zusätzlich in die Gruppe der residenten Vokabulare. Durch Nennen eines weiteren Vokabulars würde dieses nun zum transienten, könnte dann ebenfalls mit ALSO in die Gruppe der residenten übernommen werden usw. Mit dem Vokabularmechanismus kann man z. B. organisieren, daß Nutzern nur die Worte zugänglich sind, die ihrer Qualifikation entsprechen. Als zusätzliche Sicherheit (z. B. nach Fertigstellung eines Anwendungsprogramms) kann mit dem Wort SEAL (deutsch: versiegeln) die Suchreihenfolge eingefroren und der Umwählmechanismus unzugänglich gemacht werden.

2.4.3. Das Anfügevokabular

In das Anfügevokabular werden neu zu definierende Worte eingetragen. Um ein ganz bestimmtes Vokabular zum Anfügevokabular zu machen, muß es zunächst als transient erklärt werden. Ein Aufruf von DEFINITIONS bewirkt dann, daß das transiente Vokabular auch Anfügevokabular wird. Die Anzeige derjenigen Vokabulare, die aktuell zum Suchen und Anfügen vorgesehen sind, wird durch Aufruf des Wortes ORDER erreicht. Das folgende Bild zeigt als Variation zum Problem im Abschnitt 2.2.2. b), wie mittels der Vokabulartechnik zwei verschiedene Variablen mit dem gleichen Namen TEILE getrennt behandelt werden können.

```
ONLY  FORTH  DEFINITIONS(cr)_ok
VOCABULARY  HOLZ(cr)_ok
VOCABULARY  PLAST(cr)_ok
ALSO  HOLZ  DEFINITIONS(cr)_ok
VARIABLE  TEILE      0  TEILE  !(cr)_ok
PLAST  DEFINITIONS(cr)_ok
VARIABLE  TEILE      0  TEILE  !(cr)_ok
22  TEILE  +!  HOLZ  33  TEILE  +!(cr)_ok
PLAST  TEILE  @  .(cr)_22_ok
HOLZ   TEILE  @  .(cr)_33_ok
```

wird fortgesetzt

Tafel 2.1 Kurzbeschreibungen der Forthworte

Name	Stapeleffekt	Beschreibung
Speicherzugriff: Transfer zwischen Speicher und Stapel		
!	(16b addr ===>)	16b auf Adresse addr abspeichern,
@	(addr ===> 16b)	16b aus der Speicherzelle addr lesen
C!	(8b addr ===>)	8b auf Adresse addr abspeichern
C@	(addr ===> 8b)	8b aus der Speicherzelle addr lesen
+!	(w addr ===>)	zum Inhalt der Speicherzelle addr wird w addiert
2@	(addr ===> 32b)	32b ab Speicherzelle addr lesen
2!	(32b addr ===>)	32 ab Speicherzelle addr abspeichern
Behandlung von Speicherbereichen		
CMOVE	(addr1 addr2 u ===>)	ab Adresse addr1 u Bytes nach addr2 transportieren
CMOVE>	(addr1 addr2 u ===>)	ähnlich CMOVE, aber beginnend bei Adresse addr1+u−1
FILL	(addr u 8b ===>)	ab Adresse addr u Bytes mit dem Bitmuster 8b füllen
Bitoperationen		
AND	(16b1 16b2 ===> 16b3)	bitweise UND-Verknüpfung von 16b1 und 16b2 ergibt 16b3
NOT	(16b1 ===> 16b2)	16b2 ist das Einerkomplement von 16b1
OR	(16b1 16b2 ===> 16b3)	bitweise ODER-Verknüpfung von 16b1 und 16b2 ergibt 16b3
XOR	(16b1 16b2 ===> 16b3)	bitweise XOR-Verknüpfung von 16b1 und 16b2 ergibt 16b3
Definition von Worten		
:	(===>)	Beginn der Definition eines Hochwortes
;	(===>)	Beendigung der Definition eines Hochwortes
CONSTANT	(n ===>)	Konstante definieren
VARIABLE	(===>)	Variable definieren
Wörterbucharbeit		
FORGET	(===>)	Abtrennen des Wörterbuches ab folgendem Namen
Vergleiche		
0<	(n ===> ?)	flag = true, wenn Bedingung erfüllt ist
0>	(n ===> ?)	
0=	(n ===> ?)	
<	(n1 n2 ===> ?)	
>	(n1 n2 ===> ?)	
=	(n1 n2 ===> ?)	
U<	(u1 u2 ===> ?)	
D0=	(d ===> ?)	
D=	(d1 d2 ===> ?)	
D<	(d1 d2 ===> ?)	
DU<	(ud1 ud2 ===> ?)	
Entscheidungsstrukturen		
IF	(? ===>)	Anfang des *Wahr*-Zweiges
ELSE	(===>)	Anfang des *Falsch*-Zweiges
THEN	(===>)	Ende der Verzweigungskonstruktion
Schleifen mit unbestimmter Durchlaufzahl		
BEGIN	(===>)	Anfang der Schleife
UNTIL	(? ===>)	Verlassen, wenn ? = wahr
WHILE	(? ===>)	Verlassen, wenn ? = falsch
REPEAT	(===>)	Rückkehr zu BEGIN
Zählschleifen		
DO	(limit anfangswert ===>)	Eintritt in die Schleife
LOOP	(===>)	Schrittweite = 1
+LOOP	(schrittweite ===>)	Schrittweite variabel
I	(===> n)	Index der inneren Schleife
J	(===> n)	Index der äußeren Schleife, falls sie existiert
Vokabulare		
VOCABULARY	(===>)	Definitionswort für ein neues Vokabular
FORTH	(===>)	Vokabular FORTH als transient einstellen
WORDS	(===>)	Inhalt des transienten Vokabulars anzeigen
ONLY	(===>)	Suchordnung leeren
ALSO	(===>)	das transiente Vokabular wird zusätzlich resident.
SEAL	(===>)	*Versiegeln* der Suchordnung
DEFINITIONS	(===>)	das transiente Vokabular wird auch Anfügevokabular
ORDER	(===>)	Anzeigen von Suchordnung und Anfügevokabular

Fortsetzung von Seite 142

neriert wird, kann der Anwender mittels der Bits XA bzw. XB im Programm-Wort des U 80610 festlegen. Das Ausgangssignal AACK wird im Fast Cycle Mode für den Mikroprozessor U 80601 und im Slow Cycle Mode für alle K 1810 WM86-kompatiblen Prozessoren optimiert. Der Transfer Acknowledge Strobe XACK entspricht der Multibus-Spezifikation. Zur Bussteuerung ist dem Ausgang ein Tri-state-Treiber nachzuschalten. XACK wird generiert, wenn gültige Daten anliegen (Lesezyklen) oder auf dem Bus nicht mehr benötigt werden (Schreibzyklen). Es wird asynchron mit der Rücknahme des Kommandos inaktiv bzw. gar nicht erst aktiviert, wenn das Kommando zur Speicheranforderung bereits vor der Taktflanke, mit der XACK generiert werden müßte, zurückgenommen wird. Weil im asynchronen Betrieb durch den U 80610 die Daten bereits ungültig werden, bevor die CPU die Bestätigungssignale LAACK oder XACK erkannt hat, muß der Nutzer für eine Zwischenspeicherung sorgen. In synchronen Systemen erfüllt der U 80610 jedoch alle Anforderungen an die Aufstell- und Haltezeiten der Daten für den Prozessor U 80601 und für die K 1810 WM86-Familie, so daß die Daten nicht gepuffert zu werden brauchen.

Ausblick

Im Hinblick auf künftige Generationen von Prozessoren bietet der integrierte DRAM-Controller U 80610 mit seinen umfangreichen Funktionen vielfältige Einsatzmöglichkeiten. Nicht nur in leistungsstarken Personalcomputern, sondern auch in der BMSR-Technik finden sich potentielle Einsatzgebiete dieses Schaltkreises.

✉ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Applikationszentrum Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76, App. 58

Peripherieschaltkreise CIO und SCC

Marko Schmidt
VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum

CIO U 82536 DC04/U 8036 DC04
Allgemeine Beschreibung des CIO

Der CIO ist ein Zähler-/Zeitgeber- und paralleler Ein-/Ausgabeschaltkreis (Counter/Timer and Parallel I/O Unit) für 8- und 16-Bit-Mikroprozessoren.
Er besitzt zwei 8-Bit-Parallelports, ein 4-Bit-Spezialport sowie drei 16-Bit-Zähler/-Zeitgeber. Im Rahmen dieser Bedingungen sind fast alle denkbaren digitalen Funktionen, Betriebsarten und externen Zugriffe konfigurierbar.
Der CIO wird in zwei Bondvarianten angeboten, als U 82536 für den U 80601 und als U 8036 für den U 8000 (Multiplexbus) und für jeweils busverwandte Typen (K 1810 WM86, U 880, 8086, 8088, U 881–886). Die Bilder 1 und 2 enthalten die Anschlußbelegungen und die Schaltzeichen. Der CIO wird mit einem 40poligen Dual-in-line-Gehäuse gefertigt.

Aufbau des CIO

Die Baugruppen des CIO sind im Blockschaltbild im Bild 3 dargestellt.

Port A und B
Die zwei universellen 8-Bit-E/A-Ports sind identisch (Bild 4), abgesehen davon, daß Port B für den externen Zugriff auf die Zähler/Zeitgeber 1 und 2 verwendet werden kann. Jedes Port ist entweder als Parallelport mit oder ohne Handshake (Eingang, Ausgang oder bidirektional), als einzeln definierbares Bitport (Eingang oder Ausgang) oder als Steuerport programmierbar. Jedes Port besitzt eine Zeichenerkennungslogik. Sie kann so programmiert werden, daß der CIO als *Prioritätsinterruptcontroller* eingesetzt werden kann. Beide Ports können zu einem 16-Bit-E/A-Port mit Handshake gekoppelt werden.

Port C
Die Funktion des speziellen 4-Bit-Ports ist abhängig von der Betriebsart der Ports A und B. Durch Port C werden die erforderlichen Handshakesignale bereitgestellt (Bild 5). Alle Bits von Port C, die nicht als Handshakesignale benutzt werden, können als E/A-Leitungen verwendet werden oder ermöglichen den externen Zugriff auf Zähler/Zeitgeber 3. Die vier höherwertigen Bits des Port-C-Datenregisters fungieren als Schreibschutzmaske für die vier niederwertigen Bits.

Zähler/Zeitgeber
Die drei 16-Bit-Zähler/-Zeitgeber sind identisch (Bild 6). Jeder enthält einen Zähler, eine Zeitkonstante, ein Zählerstandsregister und ein Steuerregister. Bis zu vier E/A-Leitungen können als externe Zugriffsleitungen für jeden Zähler/Zeitgeber definiert werden: Zählereingang, Gateeingang, Triggereingang und Zähler-/Zeitgeberausgang. Drei verschiedene Zähler-/Zeitgeber-Ausgangstastverhältnisse sind möglich: Impuls, Einzelimpuls oder Rechteck.

Bild 1 Anschlußbelegung a) U 82536, b) U 8036

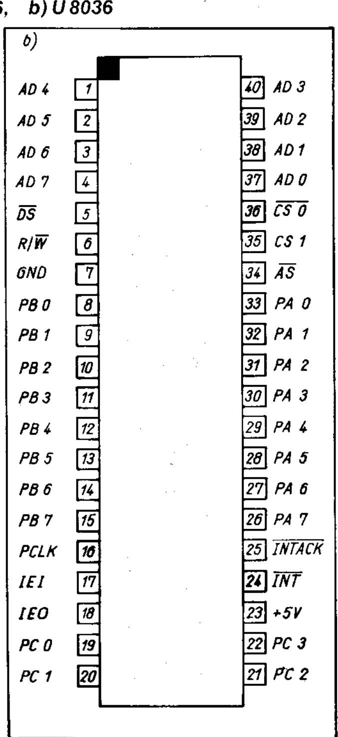


Bild 2 Schaltzeichen a) U 82536, b) U 8036

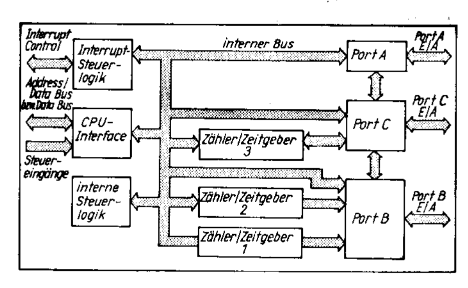


Bild 3 Blockschaltbild des CIO

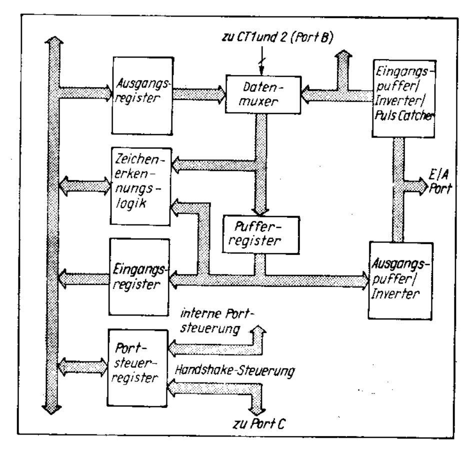


Bild 4 Aufbau der Ports A und B

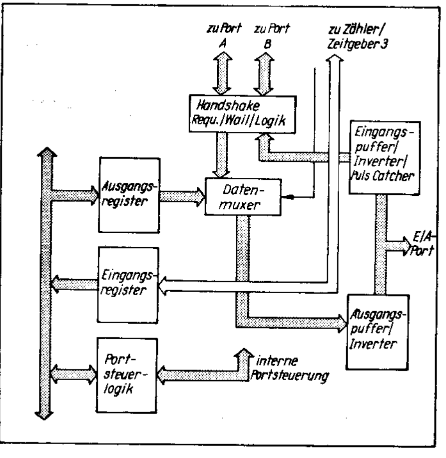


Bild 5 Aufbau von Port C

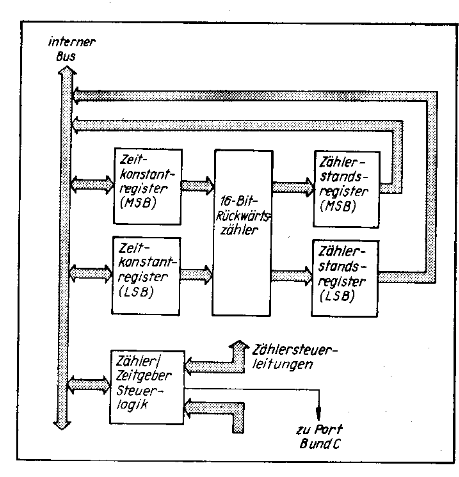


Bild 6 Aufbau der Zähler/Zeitgeber 1 bis 3

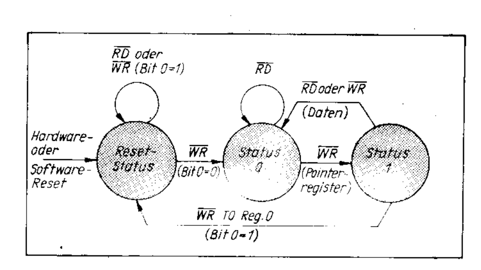


Bild 7 Zustandslogik

Interruptsteuerlogik
Die Interruptsteuerlogik ermöglicht Standard-Bus-Interrupts. Die CIO-Baugruppen bilden intern eine Prioritätskette und lassen sich extern durch IEI und IEO in eine Kette einordnen. Interruptvektoren und Interruptvektorregister mit oder ohne Status lassen eine weitgefächerte Interruptbearbeitung zu.

Programmieren des CIO

Registeradressierung des U 8036

Die Adressierung erfolgt über den gemultiplexten Adreß-/Datenbus. Es gibt zwei Möglichkeiten, die gewünschten sechs der acht Adreßbits auszuwählen. Sie werden durch das *Right Justify Address*-Bit (RJA) im *Master Interrupt Control*-Register bestimmt. Wenn RJA = 0 ist, werden die Bits AD1 bis AD6 für die Registeradresse genutzt, sonst AD0 bis AD5. Die Verschiebung ist durch die unterschiedlichen A0-Werte bei Spezial- und Normal-IN/OUTs beim U 8000 wichtig.

Registeradressierung des U 82536

Die Datenregister der Ports A, B und C sind direkt adressierbar (selektiert durch die Pins A1 und A0). Die übrigen Register sind durch eine Zwei-Schritt-Folge bei der Pinbelegung A0 = A1 = 1 zugänglich. Im ersten Schritt wird eine 6-Bit-Adresse übernommen. Im zweiten Schritt wird das selektierte Register gelesen oder beschrieben.
Der U 82536 enthält eine Zustandslogik, die festlegt, ob bei A0 = A1 = 1 ein Register adressiert oder ob auf ein vorher adressiertes Register zugegriffen wird.
Die Initialisierung des CIO geschieht in einer bestimmten Reihenfolge (Bild 7):
1. Reset
2. Reset aufheben
3. Programmieren der Funktionen
4. Setzen der entsprechenden Freigabebits.

Ein Hardware-Reset erfolgt typabhängig entweder durch gleichzeitiges Aktivieren von AS und DS oder von RD und WR. Ein Software-Reset muß durch das Einschreiben einer 0 aufgehoben werden. Die Programmierung erfolgt durch das Laden der Steuerregister mit den entsprechenden Bitmustern. Es gibt einzelne Freigabebits für die Hauptblöcke (Ports, Zähler/Zeitgeber). Vor der Freigabe können die IP-Bits nicht gesetzt werden, REQUEST und WAIT werden nicht beachtet, alle Ausgänge bleiben hochohmig, die Handshake-Signale werden ignoriert, und die Zähler/Zeitgeber können nicht getriggert werden.

Interruptsteuerung
Das *Master Interrupt Control*-Register enthält das *Master Interrupt Enable*-Bit und die Bits für das Modifizieren der drei Interruptvektoren (Vektor includes Status) sowie das Sperren der untergeordneten Prioritätskette (Disable Lower Chain).
Das *Master Configuration Control*-Register dient zur Konfigurierung und Freigabe der Zeitgeber und der CIO-Ports sowie der Verkettung der Zähler/Zeitgeber und der Ports.
Die drei *Interrupt Vector*-Register enthalten die Interruptvektoren der Ports und der Zähler/Zeitgeber, die bei einer Interrupt-Bestätigungsfolge ausgegeben werden. Wenn der Vektor modifiziert wird, enthalten zwei oder drei Bits Informationen zum Interrupt (die Ursache).
Im *OR-Priority Encoded Vector Mode* werden D3, D2 und D1 mit der Nummer des höherwertigsten Bits mit Patternübereinstimmung modifiziert (Pattern = Bitmuster).
In den übrigen Modi enthalten diese drei Bits: ORE (Output Register Empty), IRF (Input Register Full) und PMF (Pattern Match Flag).
Der Zähler-/Zeitgeber-Statusvektor enthält die Interruptquelle in D2 und D1: Zähler/Zeitgeber 3, Zähler/Zeitgeber 2, Zähler/Zeitgeber 1 oder Fehler.
Die drei *Current Vector*-Register sind für die zyklische Abfrage (Polling). Sie enthalten den Interruptvektor, der bei einer Interruptfreigabe ausgegeben worden wäre.
Die einzelnen Ports und die Zähler/Zeitgeber besitzen zusätzlich Steuer- und Statusbits für die Interruptsteuerung innerhalb der internen Prioritätskette (IP, IUS und IE). Die Prioritätsreihenfolge im CIO ist definiert: Zähler/Zeitgeber_3, Port A, Zähler/Zeitgeber_2, Port B und Zähler/Zeitgeber_1.

Programmieren der Parallelports
Der CIO hat drei Datenregister, durch die direkt auf die Ports zugegriffen werden kann. Das *Port Mode Specification*-Register und das *Port Handshake Specification*-Register definieren und steuern die Ports. *Interrupt On Match Only* steuert die Interrupterzeugung bei Pattern-Übereinstimmung.
Die *Mode Specification Bits* definieren die Betriebsart der Pattern-Match-Logik: AND, OR, OR-Priority Encoded Vector oder Disable Pattern Match. Die Kombination der Bitmuster an den Ports muß der vorgegebenen Logik genügen, damit ein *Pattern Match* ausgelöst werden kann.
LPM (Latch on Pattern Match) schaltet die Pattern-Match-Logik entweder auf Durchgang (*transparent*) oder auf Zwischenspeicher (*Latch*). DTE schaltet den *Deskew Timer*. Die *Port Handshake Specification*-Register enthalten den Typ des Handshakes, die Benutzung der REQUEST-/WAIT-Leitung und die Zeitkonstante des *Deskew-Timers*.
Die Kommando- und Statusregister (Port Command and Status) enthalten die Hauptsteuerbits und die Statusbits der Ports.
IUS wird automatisch auf 1 gesetzt, wenn in einer Interrupt-Bestätigungsfolge das entsprechende IP die anstehende Interruptanforderung mit der höchsten Priorität ist. Während IUS gesetzt ist, können gleich- und niederpriorisierte Interruptquellen keine Interruptanforderung erzeugen.
IE gibt die Port-Interruptlogik frei. Das IP-Bit wird gesetzt, wenn das Port eine Interruptbedingung erfüllt hat. INTACK setzt IP zurück. Falls IE 1 ist und keine höherpriorisierten Interrupts anliegen, wird eine Interruptanforderung erzeugt. *Interrupt Error* wird gesetzt, wenn für ein Bit-Port mit Zeichenerkennungslogik eine zweite Überdeckung auftritt, bevor die vorhergehende bestätigt wird. Falls das *Interrupt On Error*-Bit 0 ist, werden Fehler ignoriert und das Bit bleibt 0.

Handshake-Ports
Der CIO stellt vier verschiedene Handshakearten für die Ports A und B bereit: Interlock, Strobe, Puls und 3-Wire (Drei-Draht). Die Handshakelogik steuert den Datentransfer in

das Port oder aus dem Port sowie die Interrupterzeugung. Port C stellt die Handshakeleitungen bereit.
Im Interlock-Handshake muß die Handlung des CIO vom externen Gerät bestätigt werden, bevor die nächste Handlung stattfinden kann.
Im Strobe-Handshake wird das Datenwort durch die externe Logik in das Port oder aus dem Port gestrobt.
Der Puls-Handshake ist für den Anschluß mechanischer Drucker, die Daten für eine längere Zeitdauer und mit relativ großer Impulsbreite brauchen, eingerichtet. Der Interlock-Handshake wird durch das Einfügen von Zähler/Zeitgeber 3 erweitert. Beim 3-Draht-Handshake kommuniziert ein Ausgangsport mit mehreren Eingangsports gleichzeitig. Der RFD-Ausgang (Ready for Data) wird durch zwei Signale ersetzt (RFD und DAC).
Die Datenverfügbarkeit kann durch eine Zeitverzögerung verlängert werden. Dazu kann der CIO einen *Deskew Timer* in die Signalleitung DAV (Data available) einfügen. Mit DTE werden PCLK-Verzögerungszyklen (1 bis 16) zwischen der Ausgabe eines Datenbytes und der Freigabe für neue Daten (bis DAV fällt) eingefügt.

Kleines Lexikon der Mikrorechentechnik

H
wie Handshake

Zeichnung: Dahmen



Bit-Ports
Alle Ports haben je ein *Data Path Polarity*-Register (invertierend oder nichtinvertierend), ein *Data Direction*-Register (Eingang oder Ausgang) und ein *Special I/O Control*-Register [*1. Catcher* (Impulsfänger) bei Eingang, *open drain* bei Ausgang].
Es ist möglich, auf exakt zu definierende Logikbedingungen einzeln zu reagieren. Durch drei Bitmuster-Definitionsregister können die Bedingungen der Zeichenerkennungslogik für jedes einzelne Bit definiert werden. Es sind dies das *Pattern Polarity-*, das *Pattern Transition-* und das *Pattern Mask*-Register. Aus den Kombinationen dieser Register ergeben sich folgende Patterndefinitionen: Bit maskiert, beliebiger Übergang, Zustand Low, Zustand High, H/L-Flanke oder L/H-Flanke. Der CIO kann bei Übereinstimmung der Portbelegungen mit den definierten Bitmustern einen Interrupt auslösen und den auslösenden Zustand festhalten. Das Auslösen eines *Pattern-Match* ist durch logische Verknüpfungen zwischen den einzelnen Portpins möglich: OR-, AND- oder OR-Priority Encoded Vektor-Modus. Die Zeichenerkennungslogik von Bit-Ports arbeitet in zwei Grundbetriebsarten: transparent oder gelatcht. Im Latch-Betrieb wird der Zustand aller Porteingänge zu dem Zeitpunkt, da ein Match erkannt wurde, im Pufferregister zwischengespeichert und so lange gehalten, bis IP gelöscht wird. Im Transparentmodus folgen die Daten, die aus dem Port zurückgelesen werden, den Portpins. In diesem Modus weist der Interruptstatus auf das höchstpriorisierte Bit, das beim Bestätigungszyklus mit seiner Spezifikation übereinstimmt (Interruptcontroller).

Zähler/Zeitgeber
Die Arbeitsweise der drei Zähler/Zeitgeber wird durch Spezifikations-, Steuer- und Statusregister festgelegt.
In den *Counter/Timer Mode Specification*-Registern werden die externen Leitungen und die Ausgangstastverhältnisse (Impuls-, Einzelimpuls- und Rechteckimpulsausgabe) definiert.
In den *Counter/Timer Command and Status*-Registern haben IUS, IE, IP und ERR dieselbe Bedeutung wie bei den Parallelports. *Count in Process* zeigt an, ob eine Zählfolge abläuft. Es gibt je zwei Zeitkonstanten- und Zählerstandsregister, die *Counter/Timer Time Constant*-Register und die Counter/Timer Current Count-Register. Die CCR-Register folgen dem Rückwärtszähler, solange das RCC-Bit im *Counter/Timer and Status*-Register auf 1 steht, ansonsten bleibt es stehen. Dadurch kann der Wert im CCR-Register gehalten werden, bis beide Bytes von der CPU gelesen worden sind. Das RCC-Bit wird durch Lesen des niederwertigen Bytes oder durch das entsprechende Freigabebit gelöscht.

SCC U 82530 DC04/U 8030 DC04

Allgemeine Beschreibung des SCC

Die seriellen Kommunikationsschaltkreise SCC (Serial Communication Controller) U 82530 und U 8030 sind programmierbare Controller für die serielle Datenübertragung. Sie können alle wichtigen asynchronen und synchronen Standards realisieren.
Der SCC hat vier serielle Ports: zwei Sender und zwei Empfänger mit den entsprechenden Modem-Leitungen. Der Empfänger hat eine vierfache und der Sender eine zweifache Datenspeicherung. Der SCC hat einen Bitraten-Generator für jeden Kanal, eine *Digital Phase-Locked Loop* (DPLL) für die Taktgewinnung aus dem Datenstrom, einen Dekoder für NRZ-, NRZI-, FM1-, FM0- oder Manchester-Format und einen Quarz-Oszillator je Kanal. Die Betriebsarten Local Loopback und Auto Echo zur Fehlererkennung und -isolierung sind programmierbar. Die maximal mögliche Datenübertragungsrate bei 4 MHz Systemtakt beträgt:
- 1 MBit/s bei NRZ und ohne DPLL

Bild 8 Anschlußbelegung a) U 82530, b) U 8030

a)

Signal	Pin	Pin	Signal
DB1	1	40	DB0
DB3	2	39	DB2
DB5	3	38	DB4
DB7	4	37	DB6
INT	5	36	RD
IEO	6	35	WR
IEI	7	34	A/B
INTACK	8	33	CE
+5V	9	32	D/C
W/REQA	10	31	GND
SYNCA	11	30	W/REQB
RTxCA	12	29	SYNCB
RxDA	13	28	RTxCB
TRxCA	14	27	RxDB
TxDA	15	26	TRxCB
DTR/REQA	16	25	TxDB
RTSA	17	24	DTR/REQB
CTSA	18	23	RTSB
DCDA	19	22	CTSB
PCLK	20	21	DCDB

b)

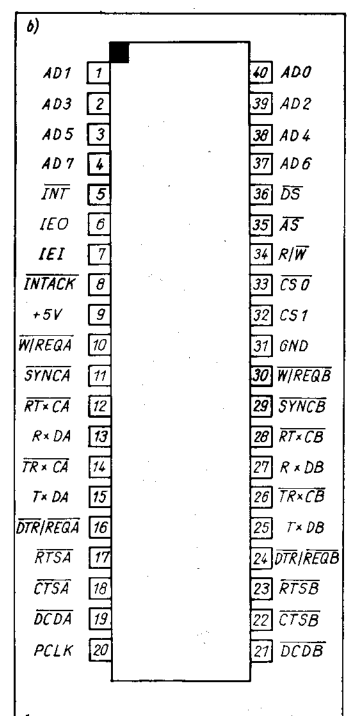


Bild 9 Schaltzeichen a) U 82530, b) U 8030

a)
SCC
4 DB7, 37 DB6, 3 DB5, 38 DB4, 2 DB3, 39 DB2, 1 DB1, 40 DB0
36 RD, 35 WR, 34 A/B, 33 CE, 32 D/C, 8 INTACK, 13 RxDA, 12 RTxCA, 18 CTSA, 19 DCDA, 27 RxDB, 28 RTxCB, 22 CTSB, 21 DCDB, 7 IEI
TRxCA 14, SYNCA 11, TRxCB 26, SYNCB 29
INT 5, TxDA 15, W/REQA 10, DTR/REQA 16, RTSA 17, TxTB 25, W/REQB 30, DTR/REQB 24, RTSB 23, IEO 6

b)
SCC
4 AD7, 37 AD6, 3 AD5, 38 AD4, 2 AD3, 39 AD2, 1 AD1, 40 AD0
35 AS, 36 DS, 34 R/W, 33 CS0, 32 CS1, 8 INTACK, 13 RxDA, 12 RTxCA, 18 CTSA, 19 DCDA, 27 RxDB, 28 RTxCB, 22 CTSB, 21 DCDB, 7 IEI
TRxCA 14, SYNCA 11, TRxCB 26, SYNCB 29
INT 5, TxDA 15, W/REQA 10, DTR/REQA 16, RTSA 17, TxDB 25, W/REQB 30, DTR/REQB 24, RTSB 23, IEO 6

● 250 KBit/s bei FM-Kodierung unter Benutzung der DPLL
● 125 KBit/s bei NRZI-Kodierung unter Benutzung der DPLL.

Der SCC wird ebenfalls in zwei Bondvarianten angeboten, als U 82530 für den U 80601 und als U 8030 für den U 8000 und für jeweils busverwandte Typen. Die Bilder 8 und 9 enthalten die Anschlußbelegungen und die Schaltzeichen. Der SCC wird mit einem 40poligen Dual-in-line-Gehäuse gefertigt.

Tafel 1 Betriebsarten des SCC

Asynchronbetrieb
– 5, 6, 7 oder 8 Bit je Zeichen – 1, 1½ oder 2 Stopbits – ungerade, gerade oder keine Parität – Taktvarianten x1, x16, x32 oder x64 – Erzeugen und Erkennen von Unterbrechungen (Breaks) – Erkennen von Paritäts-, Überlauf- oder Rahmenfehlern
byteorientierter Synchronbetrieb
– interne oder externe Zeichensynchronisierung – 1 oder 2 SYNC-Zeichen in separaten Registern – 6- oder 8-Bit-SYNC-Zeichen – automatisches Einfügen und Löschen von SYNC-Zeichen – CRC-Generierung und -Test (Cyclic Redundancy Check) – Ausschließbarkeit von Zeichen aus der CRC-Berechnung
SDLC-/HDLC-Betrieb
– Generieren und Testen von Abbruchfolgen (Abort) – automatisches Nulleinfügen und -löschen – automatisches Einfügen von SDLC-Flags in Sendesequenzen – Adreßfelderkennung – I-Feld-Residuen-Behandlung – CRC-Generierung und -Test – SDLC-Loop-Modus mit EOP-Erkennung, Schleifenankopplung und -abkopplung

Programmieren des SCC

Register

Der Registersatz des SCC besteht aus 16 Schreibregistern und 9 Leseregistern.
Die Schreibregister beinhalten die SYNC-Zeichen, die Bitrate, die zu sendenden Daten, den Interruptvektor und die Master-Interrupt-Steuerung.
Die Leseregister beinhalten die Statusregister, den Bitraten-Generator, den Interruptvektor, die empfangenen Daten und die Interrupt-Pending-Bits.

Registeradressierung des U 8030

Die Register im U 8030 werden über den Adreßbus (AD0-AD7) adressiert. Die Adresse wird mit der steigenden Flanke von AS übernommen.
Im *Shift Left*-Modus wird die Registeradresse auf AD4–AD1 plaziert, das Kanalauswahlbit A/B ist AD5. Im *Shift Right*-Modus wird A/B durch AD0 bestimmt.

Registeradressierung des U 82530

Auf die Register im U 82530 wird unter Benutzung eines Registerpointers zugegriffen. Der Zugriff erfolgt in zwei Schritten. Zum Ansprechen eines einzelnen Registers müssen die Bits des Pointers durch einen Schreibzugriff auf WR0 gesetzt werden.
Eine Lese- oder Schreiboperation mit D/C auf High greift direkt auf die Datenregister zu, unabhängig vom Zustand der Pointerbits.

Interruptsteuerung

Der SCC hat sechs Interruptquellen, drei pro Kanal. Die Quellen sind der Empfänger, der Sender und die Ext/Status-Bedingungen. Zu jeder dieser Interruptquellen gehören drei Steuer- bzw. Statusbits. Das sind *Interrupt Enable* (IE), *Interrupt Pending* (IP) und *Interrupt Under Service* (IUS). Das IP-Bit wird gesetzt, wenn die entsprechende Interruptbedingung eingetreten ist. Ein IUS wird während eines Interrupt-Anerkennungszyklus für den anliegenden Interrupt mit der höchsten Priorität gesetzt. IUS sperrt Interrupts mit niedrigerer Priorität. Am Ende einer Interrupt-Serviceroutine muß der Prozessor den Befehl *Reset Highest IUS* (WR0) geben.
Die interne Interrupt-Prioritätskette ist festgelegt: Kanal A – Kanal B und je Kanal: Empfänger-, Sender-Ext/Status-Interrupts.

Blocktransfer

Der SCC hat zur Unterstützung eines CPU-Blocktransfers und der DMA-Steuerung einen Blocktransfermodus. Dieser Modus nutzt den $\overline{W/REQ}$-Ausgang. Für die DMA-Steuerung zeigt der Requestausgang an, daß der SCC zum Datentransfer vom oder zum Speicher bereit ist.

Aufbau des SCC

Die Baugruppen des SCC sind im Blockschaltbild (Bild 10) dargestellt.

Sender

Der Sender besitzt einen 8-Bit-Datenzwischenspeicher, der vom internen Datenbus geladen wird, und ein Datenschieberegister, in das die Daten aus den Registern WR6, WR7 und WR8 (SYNC-Zeichen) geladen werden können. Zur externen Steuerung des Senders dienen die Modem-Signale RTS und DTR.

Empfänger

Der Empfänger besitzt einen Datenstack, bestehend aus drei 8-Bit-Register (FIFO), sowie ein 8-Bit-Schieberegister. Diese Konfiguration ermöglicht beim Empfang serieller Daten eine Verzögerung um drei Bytes. Das erlaubt der CPU, auch auf Interrupts infolge eines High-speed-Blocktransfers zu reagieren. Parallel zum Daten-FIFO existiert ein Fehler-FIFO, in dem Paritäts- und Rahmenfehler sowie andere Statusinformationen abgespeichert werden.
Der synchrone Empfang beginnt damit, daß im seriellen Datenstrom nach einem Bitmuster gesucht wird, das mit den programmierten SYNC-Zeichen übereinstimmt.

Bitraten-Generator

Jeder Kanal des SCC hat einen programmierbaren Bitraten-Generator. Dieser besteht aus zwei 8-Bit-Zeitkonstantenregistern und einem 16-Bit-Rückwärtszähler. Der Ausgang des BRG kann zur externen Nutzung auf TRxC gelegt werden.

Digital Phase-Locked Loop

Der SCC enthält eine interne DPLL, die dazu benutzt wird, die Taktinformation aus einem Datenstrom in NRZI- oder FM-Kodierung zurückzugewinnen.

Datendekoder

Es gibt vier durch den SCC nutzbare Datendekodierungsmethoden. Im NRZ-Kode (Non Return to Zero) wird eine 1 durch High und eine 0 durch Low dargestellt. Bei der NRZI-Kodierung (Non Return to Zero Inverted) wird die 1 durch Beibehaltung und die 0 durch Pegeländerung repräsentiert. Im FM1 (zwei Phasen für 1) findet am Anfang jedes seriellen Datenbits ein Pegelwechsel statt. Bei 1 erfolgt ein weiterer Pegelwechsel, bei 0 fehlt dieser. FM0 ist umgekehrt. Beim Mancester-Format wird in der Mitte jedes Datenbits ein Übergang erzeugt (siehe Bild 11).

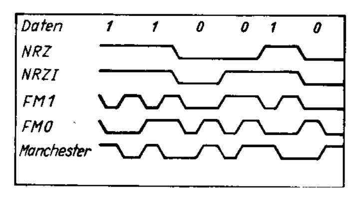


Bild 11 Datenformate des SCC

Betriebsarten des SCC

Asynchronbetrieb

Das Startbit initialisiert eine interne Taktlogik, die die synchrone Abtastung der nachfolgenden Datenbits organisiert. Die Parität wird sowohl beim Senden als auch beim Empfang berechnet. Das Stopbit setzt die Übertragungsstrecke in den Ruhezustand, das heißt, ein konstanter High-Zustand wird auf der seriellen Datenleitung gehalten bis der nächste High-Low-Übergang (Startbit) ein neues Datenzeichen ankündigt (siehe Bild 12).

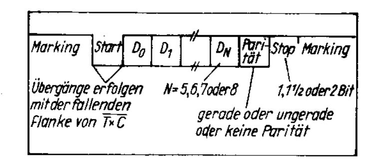


Bild 12 asynchrones Datenformat

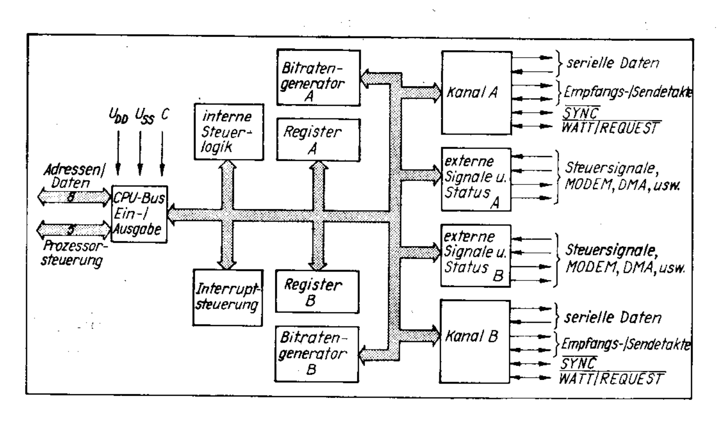


Bild 10 Blockschaltbild des SCC

Monosync-Betrieb
Monosync- und Bisync-Betrieb benötigen Taktinformationen, die gemeinsam mit den Daten übertragen werden müssen. Das geschieht entweder durch Kodierung der Daten oder über einen separaten Übertragungskanal. Mit Beginn der Übertragung setzt die CPU den Empfänger in den Hunt-Modus. Das geschieht auch dann, wenn durch eine Störung der Datenübertragung eine Neusynchronisation des Empfängers notwendig wird. Im Hunt-Modus wird der Datenstrom nach jedem ankommenden Bit mit den Bitmustern im SYNC-Zeichenregister verglichen, bis Übereinstimmung erreicht ist. Danach beginnt der Empfänger, die ankommenden Bytes in den Empfangs-FIFO zu übertragen (siehe Bild 13).

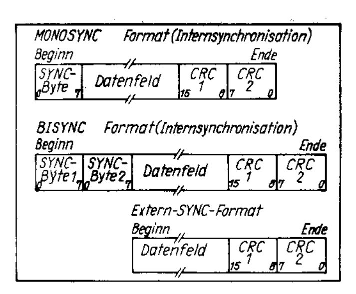


Bild 13 synchrone Formate

Bisync-Betrieb
Der Bisync-Betrieb ist dem Monosync-Betrieb sehr ähnlich, es werden jedoch zwei SYNC-Zeichen für die Synchronisation benutzt. Bisync unterstützt eine besser strukturierte Anwendung der Synchronisation durch den Einsatz von *Headern* oder *Trailern* (IBM Bisync).

Externer Synchronbetrieb
Der externe Synchronbetrieb ersetzt das SYNC-Zeichen im Datenstrom durch das Einführen eines externen SYNC-Signals, das heißt, der Beginn eines Datenfeldes wird durch Aktivierung des SYNC-Einganges angezeigt.

SDLC-Betrieb
SDLC (Synchronous Data Link Control) wird für sternförmige Datennetze genutzt. Jede Datenkommunikation benötigt mindestens zwei Stationen, eine *Primärstation* und mindestens eine *Sekundärstation*. Nicht alle Informationsübertragungen müssen durch die Primärstation veranlaßt werden.
Es werden ähnlich wie im Monosync- und im Bisync-Betrieb Synchronisationszeichen (SDLC-Flag und Adreßfeld) genutzt (Bild 14). Die beiden Flags, die den SDLC-Rahmen bilden (SDLC, EOP), dienen zum Positionieren der Adreß- und der Steuerfelder und initialisieren den Fehlertest. Das Endflag zeigt der Empfangsstation an, daß die vorher empfangenen 16 Bit das Rahmen-Testfeld bilden. Dem Endflag kann dann ein weiterer Rahmen, ein SDLC-Flag oder der Ruhezustand High folgen.

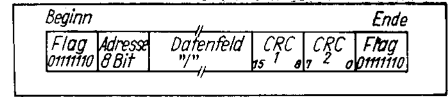


Bild 14 Sende-/Empfangs-SDLC/HDLC-Datenformat

Da im SDLC-Betrieb nicht Zeichen definierter Länge benutzt werden, sondern bitorientiert gearbeitet wird, kann das SDLC-Flag zu beliebiger Zeit erkannt werden. Um zu sichern, daß das Flag nicht zufällig in einer Datensequenz gesendet wird, muß nach der Übertragung von fünf aufeinanderfolgenden Einsen eine 0 durch den Sender eingefügt werden. Der Empfänger löscht jede 0 nach einer Folge von fünf Einsen. Sechs Einsen kennzeichnen eine Flag-Sequenz. Dadurch kann zwischen Flag und Datenfolge unterschieden werden. Die eingefügten und wieder gelöschten Nullen werden bei der CRC-Rechnung nicht berücksichtigt.
Das Adreßfeld des SDLC-Rahmens ist acht Bit lang und gibt die Adresse der Sekundärstation an, zu der die Befehle oder Daten von der Primärstation gesendet werden sollen. Das Steuerfeld ist ebenfalls acht Bit lang und wird zur Initialisierung aller SDLC-Aktivitäten genutzt.
Der SCC kann auch das HDLC-Protokoll (High Level Synchronous Data Link Communication) bedienen, es ist mit dem SDLC-Protokoll bis auf Unterschiede beim Rahmen vergleichbar.

SDLC-Loop-Modus
Der SDLC-Loop-Modus wird bei einer schleifenförmigen Vernetzung der Stationen verwendet. In einem SDLC-Loop gibt es eine als Steuerstation bezeichnete Primärstation, die den Nachrichtenfluß in der Schleife organisiert, und mehrere Sekundärstationen. Eine Sekundärstation hört den seriellen Datenstrom, der in der Schleife übertragen wird, ständig ab und gibt ihn mit einer Verzögerung von einem Bit an den Rest der Schleife weiter. Nur zu bestimmten Zeitpunkten können Sekundärstationen ihre eigenen Daten in die Schleife einspeisen.
Durch das Aussenden eines EOP (End of Poll) synchronisiert die Steuerstation die Ausgabe von Mitteilungen durch die Sekundärstationen. EOP hat die Bitfolge 11111110. Falls eine Sekundärstation Daten zu übertragen hat und ein EOP erkennt, ändert sie als erstes die letzte 1 des EOP in eine 0, bevor sie es weiterleitet. Sie ändert damit die EOP-Sequenz in eine Flag-Sequenz. Die Sekundärstation sendet nun ihre Daten in die Schleife und beendet die Übertragung mit einem EOP. Jede weitere Sekundärstation in der Schleife kann somit ihre Informationen an den Datenstrom anhängen. Die Abkopplung einer Sekundärstation von der Schleife geschieht in ähnlicher Form.

Auto Echo und Local Loopback
In der Betriebsart *Local Loopback* wird der Ausgang des Senders intern mit dem Eingang des Empfängers verbunden. Gleichzeitig werden die seriellen Daten an das TxD-Pin weitergeleitet. Das DCD-Pin (Data Carrier Detect) wird als Empfängerfreigabe und das CTS-Pin (Clear to Send) wird als Senderfreigabe auch dann ignoriert, wenn *Auto Enable* programmiert worden ist. Es ist zu beachten, daß der DPLL-Eingang und nicht der Eingang des Empfängers mit dem RxD-Pin verbunden ist. Das schließt die Nutzung der DPLL im *Local Loopback* ein.
In der Betriebsart *Auto Echo* wird intern das TxD-Pin (der Empfängereingang) direkt mit dem RxD-Pin verbunden. Dabei wird das CTS-Pin als Senderfreigabe ignoriert, und der Sendedatenausgang wird nicht angeschlossen. Diese Betriebsarten sind für Fehlererkennung und -isolierung nützlich.

✉ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Applikationszentrum Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 51076, App. 58.

TERMINE

5. Symposium „Grundlagen und Anwendung der Informatik"

WER? Technische Universität Karl-Marx-Stadt, Sektion Informatik
WANN? 6. bis 8. Februar 1990
WO? Karl Marx-Stadt
WAS?
- theoretische Grundlagen der KI
- Software-Entwicklungsumgebungen
- Compiler und Architekturen für prozedurale und logische Programmiersprachen (C, Modula-2, Lisp, Prolog u. a.)
- Fragen der Wissensdarstellung, -verarbeitung und -gewinnung
- Kopplungen von KI-Sprachen mit „klassischen" Methoden (Datenbanken, Grafik, Geometrie u. a.)
- ausgewählte Anwendungen (Computer-Algebra, Expertensysteme, CAD/CAM, CIM)

WIE? Bitte richten Sie Teilnahmemeldungen bis zum 31. Mai 1989 und Vortragsangebote bis zum 30. Juni 1989 an:
Technische Universität Karl-Marx-Stadt, Sektion Informatik, Tagungsleitung 5. Symposium, PSF 964, Karl-Marx-Stadt, 9010

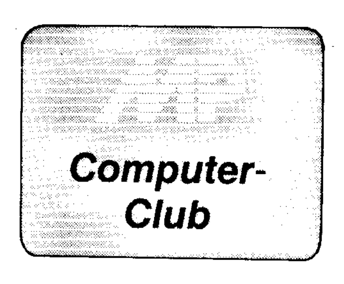
Prof. Dr. Mätzel

Fachtagung „Aufsetzmontage in der Leiterplattentechnik unter Berücksichtigung der Nacktchipverarbeitung" mit internationaler Beteiligung aus den RGW-Ländern

WER? Fachverband Elektrotechnik in der Kammer der Technik, Wissenschaftliche Sektion Technologie des elektronischen Gerätebaus
WANN? Neuer Termin: 21. bis 22. Dezember 1989 (vorher 27.–28. 9. 1989)
WO? Berlin
WAS?
- Möglichkeiten und Grenzen der Baugruppenrealisierung
- Voraussetzungen zur Einführung der Aufsetzmontage für SMDs und Nacktchips
- Verfahren und Ausrüstungen
- Fachspezifische Ausstellung

WIE? Zu diesen Schwerpunkten bitten wir um Vortragsvorschläge und Ausstellungsangebote bis zum **31. Mai 1989**. Teilnahmemeldungen sind schriftlich an folgende Anschrift zu richten: Kammer der Technik, Präsidium, Fachverband Elektrotechnik, Clara-Zetkin-Str. 115/117, Berlin, 1086; Tel. 2265285

Hoppe



Prozedurale Parameter in Turbo-Pascal

Der Pascal-Report von N. Wirth und K. Jensen /1/ sieht auch Prozeduren und Funktionen als Parameter vor, zum Beispiel

FUNCTION Y(function f(x:real):real; a,b,c:real):real;

In Turbo-Pascal ist diese Variante nicht implementiert. Bei numerischen Anwendungen kann das als Mangel empfunden werden, da strukturiert aufgebaute Programme nicht mehr *fernwirkungsfrei* gestaltet werden können.

Ein *Unterprogrammaufrufer* in Form einer External- bzw. Inline-Funktion FKTi behebt den Mangel.

Die Lösung beruht auf der Form der Übergabe von aktuellen Parametern (Argumenten) in Turbo-Pascal die via Stack erfolgt. Die Argumente werden in der Reihenfolge von links nach rechts *gekellert*, so daß das am weitesten rechts stehende Argument (Top Of Stack) übergeben wird. Ein *Unterprogrammaufrufer* wäre ein Programm, das neben allen Argumenten für die aufzurufende Funktion/Prozedur ganz rechts die Adresse dieser Funktion/Prozedur übermittelt bekommt, diese vom Stack entfernt und zur Abarbeitung der Funktion/Prozedur verzweigt.

Der externe *Unterprogrammaufrufer* besteht aus wenigen Assemblerbefehlen:

Z 80: ($E1/$E3/$E9)

```
pop  hl        ; Return-Adresse
               ; vom Stack laden
ex   (sp), hl  ; Austauschen
               ; Return-Adresse
               ; mit oberstem
               ; Integer-Wert
               ; (Adresse Funk-
               ; tion/Prozedur)
jp   (hl)      ; Abarbeitung
               ; Funktion/Proze-
               ; dur
```

8088/86: ($59/$5B/$51/$FF/$E3)

```
pop   cx   ; Return-Adresse
           ; vom Stack laden
pop   bx   ; Offset der Rou-
           ; tine vom Stack
push  cx   ; Return-Adresse
           ; zurück
jmp   bx   ; Sprung zur
           ; Routine
```

Eine allgemeingültige Anwendung zeigt das Programm FKT_Test in Bild 1.

Konkrete Anwendungen prozeduraler Parameter liegen auf dem Gebiet der numerischen Mathematik. Aber auch im Bereich der Grafik lassen sich Anwendungen finden. So könnten beispielsweise durch eine Prozedur verschiedene Linienroutinen, die für ganz spezielle periphere Geräte zugeschnitten sind, übergeben werden.

Hier sollten als Beispiele die Nullstellenbestimmung nach der Regula falsi (Bild 2) und die Integration nach der Simpsonschen Regel (Bild 3) dienen.

Die Funktion, von der im vorgegebenen Intervall die Nullstelle nach der Regula falsi bestimmt werden soll, wird unter CP/M-80 über die ADDR- und unter MS-DOS über die OFS-Funktion der Routine REGULA mitgeteilt. Dort wird mittels des *Unterprogrammaufrufers* FKT1 diese Funktion verwendet.

Literatur

/1/ Jensen, K.; Wirth, N.: PASCAL User Manual and Report. Berlin, Heidelberg, New York: Springer-Verlag 1978

/2/ Rüffer, B.: Noch ein Turbo-Tip, c't (1986) 9, S. 120

/3/ TURBO-PASCAL-Handbuch Version 3.xx. Heimsoeth Software GmbH & Co

/4/ TURBO Pascal Version 4.0 Referenzhandbuch. Heimsoeth Software GmbH & Co

Christian Hanisch

```
PROGRAM  FKT_Test; { Funktionen/Prozeduren als Parameter   }
VAR a,b,c,d,e : REAL;
    x         : CHAR;
    Adresse   : ARRAY[1..2] OF INTEGER;
                { Adress-Vektor d. Funktionen }
{ --a)-------------"Unterprogrammaufrufer"---------------- }
{ Universeller "Unterprogrammaufrufer" FKTi fuer Funktionen
                z = f(x[,y, ...])              oder Prozeduren.
Bei mehr Argumenten analoger Aufbau nach links erweitert -->
FKT1, FKT2, ... FKTi    fuer 1, 2 bis i  Parameter.
Die Adresse ADDR(<funktion>) ist immer rechts auszen als
letztes Argument (Top Of Stack) zu uebergeben, von FKTi
wird sie vom Stack entfernt und dann zu <f_addr> verzweigt.}
FUNCTION  FKT2 (y,x:REAL;f_addr:INTEGER):REAL; EXTERNAL $38;
FUNCTION  FKT1   (x:REAL;f_addr:INTEGER):REAL; EXTERNAL $38;
{ --b)-------------"Unterprogrammaufrufer"--------------- }
FUNCTION Z (x:REAL):REAL;
BEGIN Z:=x*x*12/(12+x) END;
FUNCTION quadrate(x,y:REAL):REAL;
BEGIN  quadrate:=(x*x)*(y*y)  END;
BEGIN {  Hauptprogramm: FKT_Test  }
{ --c)--------------"Unterprogrammaufrufer"--------------- }
{ Der EXTERNAL-"Unterprogrammaufrufer" wird  z.B. in die
Base Page auf 0038h bis 003Ah vor seinem Gebrauch abgelegt.
Der Bereich 0038h bis 003Ah ist fuer Debugger  DU, ZSID
u.a. reserviert.                                          }
   MEM[$003A]:=$E9;    MEM[$0039]:=$E3;     MEM[$0038]:=$E1;
{d.h. 'pop hl / ex (sp),hl / jp (hl)' ==> 0038h: "a","c","i"
           225    /     227     /  233                    }
{ --d)--------------"Unterprogrammaufrufer"--------------- }
   a:=3; b:=4; c:=9999; d:=9999; e:=9999;
   c:=quadrate(a,b);
   clrscr;
   Writeln('ohne Unterprogrammaufrufer:'#13#10#10'A=',a,
         B=',b,#13#10,'                SOLL=144 - C=',c);
   Adresse[1]:=ADDR(quadrate);
   d:=FKT2(a,b,Adresse[1]);
   Writeln(#13#10#10,'mit Unterprogrammaufrufer:',#13#10#10,
     'a=',a,'   b=',b,#13#10'                  SOLL=144 - d=',d);
   Adresse[2]:=ADDR(Z);
   e:=FKT1(b,Adresse[2]);
   Writeln(#13,#10,#10,'Funktion "e=b*b*12/(12+b)" fuer b=',
   b,#13#10'ergibt          ( SOLL=12 )          e=',e);
   Readln(x)
END.
```

Bild 1 Universeller Unterprogrammaufrufer unter CP/M-80 und Turbo-Pascal Version 3.xx

```
PROGRAM Regula_Falsi; (* TURBO PASCAL 3.0 *)
VAR K : REAL;      Flag : BOOLEAN;        x : CHAR;
FUNCTION xq(x:REAL):REAL;
BEGIN xq:=x*x+2*x-8 END;
(* ============================================== *)
FUNCTION REGULA (f_addr:INTEGER;xa,xe,eps:REAL;
                                 VAR flag:BOOLEAN):REAL;
VAR fa,fe,f0,x0 : REAL;
PROCEDURE mediator; EXTERNAL 'MEDIATOR.BIN';
FUNCTION FKT1(x:REAL;f_addr:INTEGER):REAL;
EXTERNAL mediator[0];
{ Die Datei MEDIATOR.BIN kann mit Hilfe von DEBUG auf
  folgende Art und Weise erstellt werden:
  Protokoll                Eingabe              Erlaeuterung
  ----------------------------------------------------------
  C:\debug                 debug<CR>            DEBUG rufen
  -n mediator.bin          n mediator.bin<CR>   Dateinamen
  -a                       a<CR>                Assemblerbefehl
  4D63:0100 pop cx         pop cx<CR>             Eingabe
  4D63:0101 pop bx         pop bx<CR>             Assembler-
  4D63:0102 push cx        push cx<CR>            befehle
  4D63:0103 jmp bx         jmp bx<CR>
  4D63:0105                <CR>                 beenden
  -r cx                    r cx<CR>             Reg. laden
  CX 0000
  :5                       5<CR>                5 Byte
  -w                       w<CR>                Speichern
  Writing 0005 bytes
  -q                       q<CR>                quit          }
BEGIN
  fa:=FKT1(xa,f_addr); fe:=FKT1(xe,f_addr);     {Startwerte}
  IF fa*fe<=0.0 THEN BEGIN   {Nullstelle zwischen xa und xe}
    x0:=0.5*(xa+xe);
    f0:=FKT1(x0,f_addr);
    WHILE ABS(f0)>eps DO BEGIN
      IF f0*fe>0.0 THEN BEGIN fe:=f0; xe:=x0 END
                   ELSE BEGIN fa:=f0; xa:=x0 END;
      x0:=0.5*(xa+xe);
      f0:=FKT1(x0,f_addr)
    END;
    REGULA:=x0;  flag:=TRUE
  END ELSE BEGIN REGULA:=0; flag:=FALSE END
END;
(* ============================================== *)
BEGIN  { Hauptprogramm Regula_Falsi }
  K:=REGULA(ofs(xq),1.80,3.00,0.000000000005,Flag);
  IF Flag=TRUE THEN
    Writeln('Nullstelle von x*x+2*x-8:    SOLL = 2.00'#13,
    #10'                                  IST =   ,K)
                 ELSE Writeln ('Keine Nullstelle im Bereich');
  REPEAT UNTIL keypressed
END.
```

Bild 2 Unterprogrammaufrufer am Beispiel der Regula falsi, programmtechnische Realisierung mit Turbo Pascal Version 3.xx unter MS-DOS

```
PROGRAM SIMPSON;    (* TURBO PASCAL 4.0 *)
USES crt;
VAR A,B,ERG:REAL;     n,cx,cy:INTEGER;        x:CHAR;
(* Die hier vorgestellten EXTERNAL- und INLINE-Unterpro-
   grammaufrufer duerfen nur innerhalb des Implementations-
   Teils einer UNIT oder des Hauptprogramms zusammen mit
   den zu uebergebenen Funktionen und der aufrufenden
   Prozedur/Funktion (hier SIMP) deklariert werden.
   Andere Konstellationen erfordern die Behandlung einer
   32-Bit-RETURN-Adresse oder eines far calls.             *)
```

```
(* ==================================================== *)
(*                  V a r i a n t e   1                 *)
(* -external------------------------------------------- *)

(* L mediator.obj
  Die Datei MEDIATOR.OBJ musz mit Hilfe eines kleinen Assemb-
  lerprogramms erzeugt werden. Dazu wird der folgende Assemb-
  lerquelltext mit einem geeigneten Editor in die Datei
  MEDIATOR.ASM erstellt.

              code segment byte public
                   assume cs:code
                   public fkt1
              fkt1 proc near
                   pop cx       ;Return-Adresse vom Stack
                   pop bx       ;Offset der Routine
                   push cx      ;Return-Adresse wieder rauf
                   jmp bx       ;ab zur Routine
              fkt1 endp
              code ends
                   end

   Uebersetzung:      C:\>masm mediator

FUNCTION FKT1(x:REAL;Adr_Funktion:INTEGER):REAL;
EXTERNAL;                                               *)
(* ==================================================== *)

FUNCTION Y (x:REAL):REAL;
BEGIN Y:=x*EXP(-x*x) END;

FUNCTION Z (x:REAL):REAL;
BEGIN Z:= x*x*12/(x+12) END;

(*---------------- Simpsonsche Regel -------- BEGIN ------*)

FUNCTION SIMP (n:INTEGER;a,b:REAL;f_addr:INTEGER
{Anstelle: function f(x:real):real} ):REAL;

{ ┌──────────────────────────────────────────────────────┐
  │   Integral von Lgr=a bis Ugr=b von der Funktion      │
  │   f(x) * dx = H/3*(f0 + 4*f1 + 2*f2 + 4*f3 + ... + fn)│
  │   wobei n (gerade) die Anzahl und H=(b-a)/n die Breite│
  │   der Rechtecke an den Stuetzstellen ist.            │
  │     n ::= Anzahl der Stuetzstellen                   │
  │     a ::= Untere Integralgrenze (Lgr)                │
  │     b ::= Obere Integralgrenze  (Ugr)                │
  │     f ::= Kurvenverlauf Y=f(x)                       │
  └──────────────────────────────────────────────────────┘}

VAR H,S:REAL;    I:INTEGER;

(* ==================================================== *)
(*                  V a r i a n t e   2                 *)
(* --inline-------------------------------------------- *)
(* Eine als INLINE deklarierte Funktion erledigt hier
   die Sache viel einfacher und eleganter (siehe auch
   TURBO PASCAL Referenzhandbuch S. 195):

        5B        pop bx       ; Adresse vom Stack
        FF D3     call bx      ; Prozedur rufen          *)

FUNCTION FKT1(x:REAL;Adr_Funktion:INTEGER):REAL;
INLINE($5b/$ff/$d3);
(* ==================================================== *)

BEGIN
  H:= (b-a)/n;
  S:=FKT1(a,F_addr)+FKT1(b,f_addr);
  FOR I:=1 TO n-1 DO
    IF ODD(I) THEN  S:= S + 4 * FKT1(a+I*H,f_addr)
              ELSE  S:= S + 2 * FKT1(a+I*H,f_addr);
  SIMP:=H/3*S
END;

(*---------------- Simpsonsche Regel -------- END --------*)

BEGIN                                { Hauptprogramm SIMPSON }
  N:=12;  A:=-1.50;  B:=1.50;
  clrscr;
  writeln('S I M P S O N s c h e   R e g e l':50);
  writeln;
  writeln('┌────────┬────────┬────────┬──────────────'+
  '──────────────┐');
  writeln('│ Stuetz- │ untere │ obere │      Wert de'+
  's Integrals von:     │');
  writeln('│ stellen │      Grenze     │  y=x*EXP(-x*'+
  'x)  │ y=x*x*12/(x+12) │');
  writeln('├────────┼────────┼────────┼──────────────'+
  '─────┼────────┤');
  window(1,7,80,23);
  WHILE N>0 DO BEGIN
    IF ODD(N) THEN N:=N+1;
    write('3 ',n:5,'   3 ',a:8:4,' 3 ',b:8:4,' 3 ');
    ERG:= SIMP(N,A,B,ofs(Y));
    Write(ERG:15,' 3 ');
    ERG:= SIMP(N,A,B,ofs(Z));
    Writeln(ERG:15,' 3');
    cx:=wherex; cy:=wherey;
    window(1,1,80,25);
    gotoxy(1,24);
    Write('Anzahl Stuetzstellen ( 0 fuer Ende ) : ');ClrEol;
    Read(N);
    IF n<>0 THEN BEGIN
      gotoxy(1,24);write('( <Untere Grenze> A ) : ');ClrEol;
      read(a);
      gotoxy(1,24);Write('( <Obere  Grenze> B ) : ');ClrEol;
      Read(B)
    END;
    window(1,7,80,23); gotoxy(cx,cy);
  END; { while }
END.
```

Bild 3 Unterprogrammaufrufer am Beispiel der Simpsonschen Regel, programmtechnische Realisierung mit Turbo Pascal Version 4.0 unter MS-DOS

Die Nutzung der Routinen des Basic-Interpreters

Die im Basic-Interpreter enthaltenen Arithmetikroutinen können ohne größeren Aufwand von Maschinenprogrammen genutzt werden. Dazu ist es notwendig, daß Basic schon einmal aufgerufen worden sein muß (das heißt, der Notizspeicher des Interpreters muß belegt sein). Dies kann später jedoch entfallen, indem ein Programm ab Adresse 0300H abgespeichert. Somit wird beim Laden der Notizspeicher gefüllt.
Die vier Grundrechenarten werden durch das Unterprogramm ARI realisiert. Die Werte werden an die Speicherplätze übergeben, welche durch die Allzweckregister adressiert werden. Aus dem Operationscode (ASCII in Register A) wird die Adresse der jeweiligen Arithmetikroutine ermittelt und die beiden Operanden umgeladen (OPLAD1). Der Inhalt der Adresse des Registers HL wird ins Arithmetikregister1 (03E5H) des Basic-Interpreters kopiert und der Inhalt der Adresse des Registers DE in die Register BCDE geladen. Es folgt die Ausführung der Operation. Das Ergebnis befindet sich zunächst im Arithmetikregister1 (03E5H). Dieses wird durch die Routine OPLAD2 in die Speicherzelle gebracht, die durch das Register BC adressiert wird. Da dieses Unterprogramm durch die Interpretation des Operationscodes sehr zeitaufwendig wird, ist es auch möglich die Arithmetikroutinen über die in der MP 6/87 veröffentlichten Adressen direkt aufzurufen. Dabei ist auf die Übergabe der Werte zu achten.
Durch die Subroutine INP wird die Eingabe einer Zahl über die Tastatur erreicht. Sie funktioniert ähnlich dem *INPUT* des Interpreters. Die Eingabe erfolgt immer an der aktuellen Cursorposition. Im Register HL wird die Adresse angegeben, auf welcher die konvertierte Zahl stehen soll. Zur Konvertierung wird die Interpreterroutine VALNRM genutzt. Darin werden die ASCII-Zeichen von einem durch das Register HL adressierten Speicherplatz gelesen und umgewandelt. Der Code des ersten Zeichens muß jedoch im Register A enthalten sein. Die konvertierte Zahl befindet sich nach Abschluß der Routine im Arithmetikregister1 (03E5H). Es wird auf den durch das Register HL adressierten Bereich geladen.
Zur Ausgabe von Werten auf den Bildschirm ist die Routine OUT bestimmt. Zur Rekonvertierung wird die Routine NUMKON genutzt. Die Gleichpunktzahl, welche im Arithmetikregister1 übergeben wird, wird rekonvertiert und befindet sich am Schluß in Form von ASCII-Zeichen im Print-Puffer des Interpreters. Nun folgt die Ausgabe der einzelnen ASCII-Zeichen. Es ist auch möglich, wenn der KC85/3 vorhanden, die Zeichenkette über das Unterprogramm 045H auszugeben.
Durch das Unterprogramm GPZHL wird eine Gleitpunktzahl des Basic-Interpreters, welche im Bereich von −32768 bis 32767 liegt, in eine 2-Byte-Zahl umgewandelt, die sich am Ende der Routine im Register HL (Zweierkomplement) befindet. Die Routine EPRVL3 wird im Interpreter für die Befehle **DEEK** und **PEEK** verwendet. Dazu wird die Gleitpunktzahl im Arithmetikregister1 (03E5H) übergeben. Das Ergebnis befindet sich dann im Register DE.
Das Gegenstück zu dieser Routine ist HLGPZ. Hier wird die Zahl im Register HL (Zweierkomplement) in eine Gleitpunktzahl umgewandelt. Im Register DE befindet sich die Adresse der entstehenden Gleitpunktzahl. Es wird die Routine FRE3 verwendet. Die 2-Byte-Zahl wird in den Registern B und A, welche zu BA zusammengefaßt werden, übergeben. Die Gleitpunktzahl befindet sich am Ende der Routine im Arithmetikregister1. Das Unterprogramm TEXTO ist keine Routine des Basic-Interpreters. Es dient nur zur Ausgabe einer Zeichenkette, welche mit ASCII 00 endet. Im KC85/3 kann diese durch das CAOS-Unterprogramm 045H ersetzt werden. Die im Interpreter enthaltene Subroutine zur Ausgabe einer Zeichenkette ist viel umständlicher aufgebaut und kann nicht ohne weiteres aufgerufen werden.
Die Hilfsroutine OPLSD1 dient dem Umladen der Operanden. Dabei wird der erste Operand ins Register BCDE und der zweite Operand ins Arithmetikregister1 geladen. Dabei ist zu beachten, daß zuerst der zweite Operand geladen wird, da in der Routine OPKOP das Register BCDE verändert wird. In der Hilfsroutine OPLAD2 wird das Resultat aus dem Arithmetikregister1 auf den Speicherplatz kopiert, welcher durch das Register BC adressiert wird.

Mit Hilfe des Testprogramms ist es möglich, die einzelnen Funktionen im Dialog zu testen.

%HGLPZ nnnn
PE: nnnn – 2-Byte-Hexadezimalzahl
PA: "GPZ=" – Gleitpunktzahl

%GPZHL
PE: "GPZ=" – Gleitpunktzahl
PA: "HL=" – 2-Byte-Hexadezimalzahl

%INPUT
PE: "GPZ=" – Gleitpunktzahl
PA: "GPZ=" – 4-Byte-Gleitpunktzahl in hexadezimaler Darstellung

%OUTPUT nnnn
PE: nnnn – Adresse der Gleitpunktzahl
PA: "GPZ=" – Gleitpunktzahl

%ARI
PE: "Op.1=" – Operand1
"Op.2=" – Operand2
"Op.code=" – Operationsart
PA: "Res.=" – Resultat

Andreas Zierott

Die Bilder zu diesem Beitrag finden Sie auf den beiden folgenden Seiten.

```
6000                    ORG     0000H
0000            ;************************************;
0000            ;   Routinen   zur   Nutzung   der   ;
0000            ;   BASIC- Interpreterarithmetik     ;
0000            ;------------------------------------;
0000            ;   A. Zierott           5.9.1987    ;
0000            ;************************************;
0000            ;
0000            PV1     EQU     0F003H
0000            CURSO   EQU     0B7A0H  ;Cursorposition
0000            VALNRM  EQU     0D7A1H  ;ASC => GPZ
0000            WRA1    EQU     003E5H  ;Rechenregister
0000            NUMKON  EQU     0D834H  ;GPZ => ASC
0000            INTPRB  EQU     003EAH  ;Print-Puffer
0000            OPLAD   EQU     0D6EEH  ;(HL)=> BCDE
0000            OPKOP   EQU     0D6DDH  ;(HL)=> WRA1
0000            ADD4    EQU     0D46CH  ;Addtion
0000            ADD5    EQU     0D46FH  ;Subtraktion
0000            MUL1    EQU     0D59AH  ;Multiplikation
0000            DIV1    EQU     0D5F5H  ;Division
0000            EPRVL3  EQU     0C96FH  ;GPZ => DE
0000            FRE3    EQU     0D0B1H  ;AB  => GPZ
0000            OP1     EQU     00134H  ;Adr. Operand1
0000            OP2     EQU     00138H  ;Adr. Operand2
0000            RES     EQU     0013CH  ;Adr. Resultat
0000            ;Add. / Sub. / Mul. / Div.
0000            ;====================================
0000            ;PE: HL - Adresse des 1. Operanden
0000            ;    DE - Adresse des 2. Operanden
0000            ;    BC - Adresse des Resultates
0000            ;    A  - Operationscode (+-*/)
0000            ;PA: Resultat in (BC)
0000 F5         ARI     PUSH    AF
0001 C5                 PUSH    BC
0002 D5                 PUSH    DE
0003 E5                 PUSH    HL
0004 21B700             LD      HL,CODTAB
0007 11BC00             LD      DE,ADRTAB
000A 46                 LD      B,M
000B 0E00               LD      C,0
000D 23         ARI0    INC     HL
000E BE                 CP      M
000F 2808               JR      Z,ARI2
0011 0C                 INC     C
0012 10F9               DJNZ    ARI0
0014 E1         ARI1    POP     HL
0015 D1                 POP     DE
0016 C1                 POP     BC
0017 F1                 POP     AF
0018 C9                 RET
0019 2600       ARI2    LD      H,0
001B 69                 LD      L,C
001C 29                 ADD     HL,HL
001D 19                 ADD     HL,DE
001E 4E                 LD      C,M
001F 23                 INC     HL
0020 46                 LD      B,M
0021 E1                 POP     HL
0022 D1                 POP     DE
0023 D5                 PUSH    DE
0024 E5                 PUSH    HL
0025 C5                 PUSH    BC
0026 CD9B00             CALL    OPLAD1
0029 E1                 POP     HL
002A CDB600             CALL    CAHL
002D E1                 POP     HL
002E D1                 POP     DE
002F C1                 POP     BC
0030 F1                 POP     AF
0031 1872               JR      OPLAD2
0033            ;Eingabe einer GPZ
0033            ;====================================
0033            ;PE: HL - Adresse der GPZ
0033            ;PA: GPZ in (HL)
0033 F5         INP     PUSH    AF
0034 C5                 PUSH    BC
0035 D5                 PUSH    DE
0036 E5                 PUSH    HL
0037 ED5BA0B7           LD      DE,(CURSO)
003B CD03F0             CALL    PV1
003E 32                 DEFB    32H
003F E5                 PUSH    HL
0040 CD03F0             CALL    PV1
0043 17                 DEFB    17H
0044 E1                 POP     HL
0045 7E                 LD      A,M
0046 CDA1D7             CALL    VALNRM
0049 D1                 POP     DE
004A D5                 PUSH    DE
004B 21E503             LD      HL,WRA1
004E 010400             LD      BC,4
0051 EDB0               LDIR
0053 E1                 POP     HL
0054 D1                 POP     DE
0055 C1                 POP     BC
0056 F1                 POP     AF
0057 C9                 RET
0058            ;Ausgabe einer GPZ
0058            ;====================================
0058            ;PE: HL - Adresse der GPZ
0058            ;PA: Zahl auf Bildschirm
0058 F5         OUT     PUSH    AF
0059 C5                 PUSH    BC
005A D5                 PUSH    DE
005B E5                 PUSH    HL
005C 11E503             LD      DE,WRA1
005F 010400             LD      BC,4
0062 EDB0               LDIR
0064 CD34D8             CALL    NUMKON
0067 21EA03             LD      HL,INTPRB
006A CD9100             CALL    TEXTO
006D 18A5               JR      ARI1
006F            ;GPZ ==> 2-Bytehexadezimalzahl wandeln
006F            ;====================================
006F            ;PE: HL - Adresse der GPZ
006F            ;PA: HL - Hexadezimalzahl
006F D5         GPZHL   PUSH    DE
0070 C5                 PUSH    BC
0071 F5                 PUSH    AF
0072 EB                 EX      DE,HL
0073 CD9B00             CALL    OPLAD1
0076 CD6FC9             CALL    EPRVL3
0079 EB                 EX      DE,HL
007A F1                 POP     AF
007B C1                 POP     BC
007C F1                 POP     AF
007D C9                 RET
007E            ;2-Bytehexadezimalzahl ==> GPZ wandeln
007E            ;====================================
007E            ;PE: HL - Hexadezimalzahl
007E            ;    DE - Adresse der entstehenden GPZ
007E            ;PA: GPZ in (DE)
007E E5         HLGPZ   PUSH    HL
007F C5                 PUSH    BC
0080 F5                 PUSH    AF
0081 D5                 PUSH    DE
0082 7C                 LD      A,H
0083 45                 LD      B,L
0084 CDB1D0             CALL    FRE3
0087 C1                 POP     BC
0088 C5                 PUSH    BC
0089 CDA500             CALL    OPLAD2
008C D1                 POP     DE
008D F1                 POP     DE
008E C1                 POP     BC
008F E1                 POP     HL
0090 C9                 RET
0091            ;Ausgabe einer Zeichenkette
0091            ;====================================
0091            ;PE: HL - Adresse
0091            ;PA: HL - Ende+1
0091            ;    A  - 00
0091            ;    Zeichenkette auf Bildschirm
0091 7E         TEXTO   LD      A,M
0092 23                 INC     HL
0093 B7                 OR      A
0094 C8                 RET     Z
0095 CD03F0             CALL    PV1
0098 00                 NOP
0099 18F6               JR      TEXTO
009B            ;Operanden umladen
009B            ;====================================
009B            ;PE: HL - Adresse des 1. Operanden
009B            ;    DE - Adresse des 2. Operanden
009B            ;PA: BCDE - 1. Operand
009B            ;    WRA1 - 2. Operand
009B E5         OPLAD1  PUSH    HL
009C EB                 EX      DE,HL
009D CDDDD6             CALL    OPKOP   ;(HL).4=>WRA1
00A0 E1                 POP     HL
00A1 CDEED6             CALL    OPLAD   ;(HL).4=>BCDE
00A4 C9                 RET
00A5            ;Resultat umladen ( WRA1=>(BC) )
00A5            ;====================================
00A5            ;PE: BC - Adresse des Resultates
00A5            ;    WRA1 - Resultat
00A5            ;PA: Resultat in (BC)
00A5 F5         OPLAD2  PUSH    AF
00A6 C5                 PUSH    BC
00A7 D5                 PUSH    DE
00A8 E5                 PUSH    HL
00A9 21E503             LD      HL,WRA1
00AC 50                 LD      D,B
00AD 59                 LD      E,C
00AE 010400             LD      BC,4
00B1 EDB0               LDIR
00B3 C31400             JP      ARI1
00B6            ;Simulation eines      CALL     (HL)
00B6            ;====================================
00B6 E9         CAHL    JP      (HL)
00B7 04         CODTAB  DEFB    4
00B8 2B2D2A2F           DEFM    '+-*/'
00BC 6FD4       ADRTAB  DEFW    ADD5
00BE 6CD4               DEFW    ADD4
00C0 9AD5               DEFW    MUL1
00C2 F5D5               DEFW    DIV1
```

```
ERRORS: 0000
```

```
6000                    ORG     0BA00H
BA00            ;************************************;
BA00            ;  Testprogramm zur Nutzung der      ;
BA00            ;  BASIC- Interpreterarithmetik      ;
BA00            ;------------------------------------;
BA00            ;  A. Zierott           5.9.1987     ;
BA00            ;************************************;
BA00            ;
BA00            ARI     EQU     00000H
BA00            INP     EQU     00033H
BA00            OUT     EQU     00058H
BA00            GPZHL   EQU     0006FH
BA00            HLGPZ   EQU     0007EH
BA00            PV1     EQU     0F003H
BA00            OP1     EQU     00134H
BA00            OP2     EQU     00138H
BA00            RES     EQU     0013CH
BA00 7F7F               DEFW    7F7FH
BA02 484C4750           DEFM    'HLGPZ'
BA07 01                 DEFB    1
BA08 113401             LD      DE,OP1
BA0B CD7E00             CALL    HLGPZ
BA0E EB                 EX      DE,HL
BA0F CD5800             CALL    OUT
BA12 C3D7BA             JP      CRLF
BA15 7F7F               DEFW    7F7FH
BA17 47505A48           DEFM    'GPZHL'
BA1C 01                 DEFB    1
BA1D CDCBBA             CALL    GPZOUT
BA20 213401             LD      HL,OP1
BA23 CD3300             CALL    INP
BA26 CD6F00             CALL    GPZHL
BA29 CD03F0             CALL    PV1
BA2C 23                 DEFB    23H
BA2D 484C3D             DEFM    'HL='
BA30 00                 NOP
BA31 CD03F0             CALL    PV1
BA34 1A                 DEFB    1AH
BA35 C3D7BA             JP      CRLF
BA38 7F7F               DEFW    7F7FH
BA3A 494E5055           DEFM    'INPUT'
BA3F 01                 DEFB    1
BA40 CDCBBA             CALL    GPZOUT
BA43 213401             LD      HL,OP1
BA46 CD3300             CALL    INP
BA49 213701             LD      HL,OP1+3
BA4C 0604               LD      B,4
BA4E CDCBBA             CALL    GPZOUT
BA51 7E         INPU1   LD      A,M
BA52 2B                 DEC     HL
BA53 CD03F0             CALL    PV1
BA56 1C                 DEFB    1CH
BA57 10F8               DJNZ    INPU1
```

```
BA59 187C                JR     CRLF
BA5B 7F7F                DEFW   7F7FH
BA5D 4F555450            DEFM   'OUTPUT' ;aus Adresse
BA63 01                  DEFB   1
BA64 CDCBBA              CALL   GPZOUT
BA67 CD5800              CALL   OUT
BA6A 186B                JR     CRLF
BA6C 7F7F                DEFW   7F7FH
BA6E 415249              DEFM   'ARI'
BA71 01                  DEFB   1
BA72 CD03F0              CALL   PV1
BA75 23                  DEFB   23H
BA76 4F702E31            DEFM   'Op.1='
BA7B 00                  NOP
BA7C 213401              LD     HL,OP1
BA7F CD3300              CALL   INP
BA82 CD03F0              CALL   PV1
BA85 23                  DEFB   23H
BA86 4F502E32            DEFM   'OP.2='
BA8B 00                  NOP
BA8C 213801              LD     HL,OP2
BA8F CD3300              CALL   INP
BA92 CD03F0              CALL   PV1
BA95 23                  DEFB   23H
BA96 4F702E63            DEFM   'Op.code='
BA9E 00                  NOP
BA9F CD03F0              CALL   PV1
BAA2 04                  DEFB   4
BAA3 F5                  PUSH   AF
BAA4 CD03F0              CALL   PV1
BAA7 00                  NOP
BAA8 CD03F0              CALL   PV1
BAAB 2C                  DEFB   2CH
BAAC F1                  POP    AF
BAAD 213401              LD     HL,OP1
BAB0 113801              LD     DE,OP2
BAB3 013C01              LD     BC,RES
BAB6 CD0000              CALL   ARI
BAB9 CD03F0              CALL   PV1
BABC 23                  DEFB   23H
BABD 5265732E            DEFM   'Res.='
BAC2 00                  NOP
BAC3 213C01              LD     HL,RES
BAC6 CD5800              CALL   OUT
BAC9 180C                JR     CRLF
BACB CD03F0    GPZOUT    CALL   PV1
BACE 23                  DEFB   23H
BACF 0D0A                DEFW   0A0DH
BAD1 47505A3D            DEFM   'GPZ='
BAD5 00                  NOP
BAD6 C9                  RET
BAD7 CD03F0    CRLF      CALL   PV1
BADA 2C                  DEFB   2CH
BADB C9                  RET

ERRORS: 0000
```

Turbo-Pascal-Routine für den A 7100

Die im folgenden vorgestellte Turbo-Pascal-Routine beseitigt einen Schönheitsfehler des A 7100 oder besser: seines Betriebssystems SCP 1700.
Da in der Routine direkt auf die Hardware des Rechners zugegriffen wird, ist sie *nur* auf dem A 7100 verwendbar!
Normalerweise kann unter dem Betriebssystem CP/M durch das Steuerzeichen Control-G (ASCII-Code 7) ein Signalton ausgelöst werden. Obwohl der A 7100 über eine eingebauten Piezo-Signalgeber verfügt, wird durch das Betriebssystem SCP 1700 die Signaltonausgabe nicht unterstützt.
Laut /1/ S. 92 wird der Signalgeber durch Setzen des Bit 6 im PPI-Port C für etwa 30 ms ausgelöst. Durch die Prozedur BEEP wird der Signalgeber n-mal im Abstand von m Millisekunden aktiviert. Liegt die festgelegte Zeitkonstante unter der Zeitdauer eines einzelnen Signaltones, so ergibt sich ein zusammenhängender, langer Ton; ist sie länger, so entsteht eine auffällige, intermittierende Tonfolge, die aus n Einzeltönen besteht.
Im Programm SIGNAL werden die verschiedenen Möglichkeiten demonstriert.
Da die Zeitdauer einer mit der Standard-Prozedur DELAY erzeugten Pause implementationsabhängig ist, kann eventuell eine experimentelle Anpassung der Werte für die Pausenzeit erforderlich sein.

Literatur

/1/ Betriebsdokumentation **A 7100, Band 1:** Rechner und Gerät. VEB Robotron-Elektronik Dresden

B. Matzke

```
program signal;
procedure beep(n,m:byte);
  const  c = $0cc;                     {Portadresse          }
  var  i  :  byte;
  begin
    for  i:=1  to  n  do  begin        {n  Einzeltöne        }
      port[c]:=port[c]  and  191;      {löschen Bit 6        }
      port[c]:=port[c]  or  64;        {setzen Bit 6         }
      delay(m); end;                   {m  Millisekunden Pause }
  end;

begin
    beep(10,20); delay(1000);          {ein langer Ton       }
    beep(10,50); delay(1000);          {intermittierender Ton}
    beep(2,255);                       {zwei kurze Einzeltöne}
end.
```

Wegbereiter der Informatik



CHARLES BABBAGE

***1792 Teignmouth, †1871 London**

Charles Babbage war Mathematikprofessor in Cambridge und Mitglied der Royal Society. Er verfaßte ungewöhnlich genaue Logarithmentafeln für den praktischen Gebrauch. Da er in den damals benutzten Tabellenwerken wiederholt zahlreiche Fehler feststellte, kam er auf die Idee, eine Rechenmaschine zur Herstellung neuer und zur Prüfung schon vorhandener mathematischer Tafeln zu entwerfen. Das grundlegende Prinzip, das in dieser Maschine zur Anwendung kommen sollte, war die Differenzenmethode: Zur Berechnung von Funktionen an bestimmten Zwischenstellen sollten geometrische Progressionen mit konstanter n-ter Differenz (bis zu n = 5) schrittweise aufgerechnet und die Ergebnisse ausgedruckt werden. Babbage nannte seine Maschine *Difference Engine*.
Im Jahre 1822 wurde ein erster Prototyp fertiggestellt: Er hatte drei Achsen mit jeweils fünf Ziffernrädern und konnte Zahlenfolgen mit konstanter 2. Differenz (bis 99999) berechnen. Diese Konstruktion erregte offenbar große Bewunderung, was den Erfinder veranlaßte, sogleich eine wesentlich größere Maschine zu konzipieren. Vor allem wollte Babbage die Rechengeschwindigkeit steigern: Die neue Maschine sollte 44 Rechenschritte pro Minute ausführen. Dazu entwickelte er Mechaniken, die in allen Ziffernstellen eine gleichzeitige Ausführung der Addition und des Zehnerübertrags bewerkstelligten. Da er den finanziellen Aufwand nicht allein tragen konnte, ersuchte er die die Regierung um Unterstützung, die ihm – wohl auf ein Gutachten der Royal Society hin – auch gewährt wurde: Man richtete ihm eine eigene Werkstatt ein und engagierte einige Mechaniker.
Die Arbeit an der Difference Engine Nr. 2 nahm viel Zeit in Anspruch, mußte doch Babbage erst die Vorrichtungen und Werkzeugmaschinen entwerfen, welche die präzisen Verzahnungen der Ziffernräder gewährleisteten. Ab 1829 häuften sich jedoch die finanziellen Schwierigkeiten, die Arbeiten gerieten immer mehr ins Stocken und mußten 1842 ganz eingestellt werden, weil die Regierung wegen unbefriedigender Ergebnisse die Weiterfinanzierung einstellte. Die unvollendete Difference Engine Nr. 2 kam mit allen Konstruktionsplänen nach London in das Museum des King's College.
Durch seinen Mißerfolg offenbar keineswegs entmutigt, wandte sich Babbage ab 1833 einem weit umfangreicheren Projekt zu, der *Analytical Engine*. Diese sollte ein programmgesteuerter Universalrechner mit folgenden Baugruppen werden:
– Rechenwerk für dezimale Zahlendarstellung mit Schaltgetrieben zur Realisierung einer Rechenablauf-Steuerung
– Speicher für eintausend 50stellige Zahlen
– Eingabewerk für Zahlen, Variablen und Rechenvorschriften, wobei die von Joseph Marie Jacquard 1804–08 erfundenen Lochkarten verwendet werden sollten
– Druckwerk.
Mit dieser Konzeption war Babbage seiner Zeit weit voraus – eigentlich zu weit voraus: Die ungenügenden Fertigungstechniken verhinderten, daß seine Pläne damals verwirklicht werden konnten. Sein Sohn H. P. Babbage vollendete später einen Teil der arithmetischen Einheit und berechnete zur Demonstration der Brauchbarkeit eine Tafel der Zahl π.
Erst 100 Jahre später gelang es dann Konrad Zuse, übrigens ohne Kenntnis der Babbageschen Pläne, den ersten programmgesteuerten Rechenautomaten aufzubauen.

Dr. Klaus Biener

Softwareentwicklungs-arbeitsplatz SEAP 16 Version 2.0

Der im August 1988 in der Mikroprozessortechnik angebotene Softwareentwicklungsarbeitsplatz für 16-Bit-PCs (Version 1.2) zur Unterstützung der Softwareentwicklung auf der Grundlage des Betriebssystems DCP wurde bereits vielfach nachgenutzt. Die aus den Komponenten *Handbuch 16-Bit-PC, Kommandoprozeduren* und *Hilfsprogramme* bestehende Version 1.2 wird folgendermaßen erweitert:

- Einbeziehung des ESER-PCs EC 1834
- Ausstattung aller Kommandoprozeduren mit HELP-Informationen
- Zusätzliche Unterstützung vo n Assembler sowie der Programmiersprachen C und Turbo-Pascal 4.0
- Erweiterung bzw. Neuaufnahme des Punktes *Sicherungstechnologie*
- Aufbau eines maschinell unterstützten Informationssystems für die Arbeit mit SEAP 16
- Entwicklung von menügesteuerten Lösungen für auswahlorientierte Aufgaben. Dazu gehören z. B. *Diskette formatieren* und *Drucker einstellen.*
- Bereitstellung von Informationen über neue Softwareprodukte und Hinweise zu ihrer Anwendung.

Die Version 2.0 ist voraussichtlich ab 5/89 verfügbar.

VEB DVZ Halle, Abteilung FP, Block 081, Halle-Neustadt, 4090

Gruhl

Programmdokumentation für Turbo-Pascal

Die vorhandene Software zur Dokumentation von Turbo-Pascal-Programmen ist mangelhaft, da der Quelltext zwar ausgedruckt werden kann, aber selbst die einfachste Aufbereitung nur selten möglich ist. Aufgabe der Software **PLIST+** ist es, einen Quelltext so zu dokumentieren und zu analysieren, daß er auch nach einem längeren Zeitraum innerhalb kurzer Frist erschlossen werden kann. Ausgehend von dem durch PLIST (siehe CHIP-special-Turbopascal Nr. 1, Vogel-Verlag Würzburg 1985) erreichten Stand wurde das Programm entwickelt, das folgende Funktionen wahlweise realisiert:

- Programmlisting mit Zeilennummerierung, Angabe der Herkunft (Main- oder Includefile), der Verschachtelungstiefe und Markierung der reservierten Wörter
- Annahme von Drucksteuerzeichen
- Zusammendruck von zusammengehörigen Blöcken (z. B. Prozeduren)
- Schriftart MINITYPE auf K 6313-kompatiblen Druckern (ca. 120 Zeilen pro A4-Seite)
- Eindruck der Includefiles
- Crossreferenzliste nach dem Vorbild von PLIST
- Übersicht zur statischen Struktur, die die Prozeduren und Funktionen der Reihenfolge nach ihrer globalen und lokalen Wichtung auflistet und das Herkunftsfile angibt
- ausreichende Hilfestellung und einfache Benutzersteuerung durch Dialogsteuerung und Standardannahmen.

Lauffähige Varianten existieren auf PC 1715 unter SCP, CP/M und CP/A, auf A 7100 unter SCP 1700 und CP/K sowie auf dem Schneider PC 1512.

Technische Hochschule Ilmenau, Sektion Elektrotechnik, PSF 327, Ilmenau, 6300

Weinmeister

EPROMer für Kompatible

Unter dem Aspekt der Notwendigkeit eines IBM-kompatiblen EPROM-Programmiergerätes erfolgte die Entwicklung eines EPROMers mit der Anschlußkompatibilität zu fast allen Rechnern. Voraussetzung ist eine parallele Schnittstelle (Centronics, IFSP o. ä.). Die Datenübertragung erfolgt seriell, so daß insgesamt nur vier Leitungen erforderlich sind. Zur Anwendung kommen ausschließlich DDR-Bauelemente, wobei die gesamte Schaltung, einschließlich Netzteil, auf einer zweiseitigen Leiterkarte von etwa der Größe einer EURO-Karte Platz findet. Die Steuerung des EPROMers erfolgt über in Turbo-Pascal 3.0 geschriebene Software. Dadurch ist der Betrieb des Gerätes leicht für andere Rechner modifizierbar. Zur Zeit arbeitet der EPROMER an einem Schneider PC 1640 (Druckerschnittstelle LPT1) und läßt die Programmierung der Typen 2716 bis 27256 einschließlich der A-Typen zu.

AdW, ZOS, Abt. 9.3, Rudower Chaussee 6, Berlin, 1199; Tel. 6744043

Rohloff

dBase II-POWER

Das Programm wurde für den AC 7100 unter SCP 1700 entwickelt, um den Bedienern mit nur geringen dBase II-Kenntnissen die Pflege der von ihnen über andere Programme genutzten Dateien zu ermöglichen sowie die Arbeit mit Disketten so zu gestalten, daß das Datenbankbetriebssystem nicht verlassen werden muß. Das Programm hat folgende Merkmale:

- Arbeit mit beliebigen Dateien: Wechsel des Laufwerkes, Anzeige des Directories (mit Ausgabemaske) sowie Umbenennen und Löschen von Dateien (Löschschutz mittels Kennwort).
- Arbeit mit dBase II-Dateien: Datei sortieren/indizieren, Inhalt einer Datei anzeigen/verändern, Löschmarkierungen setzen/aufheben, neue Sätze anhängen und das Erstellen einer Sicherheitskopie der bearbeiteten Datei. Außerdem werden von der im Zugriff befindlichen Datei im Hauptmenü stets die Satzlänge, die Anzahl der Sätze sowie der auf der Diskette benötigte Speicherplatz angezeigt (wahlweise auch die Struktur). Weiterhin können die Belegungen der Funktionstasten im SCP 1700 für die Dauer der dBase II-Sitzung verändert werden. Es ist weiterhin möglich, die Feldnamen und -typen einer Datei zu ändern, ohne daß deren Inhalt zerstört wird oder die Datei ausgelagert werden muß.

VEB Landmaschinenbau Torgau, Abt. Produktionsorganisation, Prager Straße 1, Torgau, 7290; Tel. 59384

Wötzel

Maus-Programmierung und Grafik am PC 1512

Für das Programmierwerkzeug Turbo-Pascal 4.0 (Borland Inc.) wurden zwei allgemein verwendbare Sammlungen von Funktionen, Prozeduren, Konstanten und Typen (Units) erarbeitet, die für Programmierer von Grafikanwendungen interessant sind.

Als Ergänzung zu den Standard-Units wurde eine Unit **MOUSE** erarbeitet, die dem Anwendungsprogrammierer die Grundfunktionen der allgemein üblichen (microsoft-kompatiblen) Betriebssystemschnittstelle zur Maus bereitstellt. Die Prozedur- und Funktionsdeklarationen sind weitgehend analog zu einer üblichen Modula-2-Funktionsbibliothek für den gleichen Zweck. Weiterhin wurde für den weit verbreiteten Personalcomputer PC 1512 mit der Unit **PC1512** eine Möglichkeit geschaffen, 16farbige Bildschirmgrafiken mit 640 × 200 Bildpunkten zu erzeugen, die bisher bekannt gewordene Lösungen bei weitem übertrifft. Sie ist als optionale und vollkompatible Erweiterung der Standard-Unit GRAPH konzipiert. Alle Funktionen der Standard-Unit wirken unverändert weiter. Wird jedoch als Hardware ein PC 1512 vorgefunden, dann sind durch Nutzung der konkreten Hardwareeigenschaften dieses PCs sowie geschickte Emulation innerhalb der Unit PC1512 anwendungsprogrammseitig alle in der Standard-Unit GRAPH deklarierten Funktionen auch in zusätzlichen Grafikmodi verfügbar.

Friedrich-Schiller-Universität Jena, Sektion Technologie für den WGB, Technikum LAURA, Ernst-Thälmann-Ring 32, Jena, 6900; Tel. 8224914

Ginter

Ergänzungssoftware für SCP-Systeme

Es werden Programme für 8-Bit-Rechner mit CP/M-kompatiblen Betriebssystemen vorgestellt, die als Ergänzungssoftware zu vorhandener Standardsoftware vorgesehen sind.

1. **ASSA:** *kombiniertes Debug-/Absolutassemblersystem*
Absolutassembler mit integrierten Debugfunktionen (LOAD-/SAVE-Funktion, Speichermanipulationen, Prüfsummen) für eine effektive Aufbereitung des assemblierten Maschinenkodes zu COM-Dateien. Der Assembler verarbeitet Z 80-Mnemonik und den größten Teil der MRES-Notation.
2. **EDITM:** *Quellkode-Editor*
Zugehöriger Editor zu ASSA. Die Quellkodedateien können jedoch auch mit TP editiert werden.
3. **UPS:** *Unterprogrammsystem für Gerätebedienung*
Bereitstellung von Unterprogrammen für Bildschirm-/Tastatur- und Druckerbedienung sowie für eine effektive Abwicklung der Arbeit mit sequentiellen SCP-Dateien auf dem Niveau der Anwendungsprogrammierung. Der Programmierer wird u. a. von folgenden Problemen entlastet: Programmierung der Eingabe des Filenamens, der Fehlerbehandlung, der OPEN-/CLOSE-Aufrufe.
4. **TDLS:** *Spezial-Basicinterpreter*
Anpassung des aus der Literatur bekannten TDL-Basic an SCP, CP/M. Zusätzlich wurden LOAD- und SAVE-Funktionen von/zur Diskette integriert. Die Zeicheneingabe kann wahlweise über Tastatur oder durch Spracheingabe erfolgen (ESE K 7821, Robotron).
5. **PROM8:** *EPROM-Programmierroutine für U 555 und U 2716*
Das Programm beinhaltet neben den Grundfunktionen EPROM-Lesen/-Programmieren zahlreiche Debugfunktionen.

VEB EAB/ZFT, Abt. RC4, Rhinstraße 100, Berlin, 1140; Tel. 5509531/App. 29

Dr. Mertins

80 Zeichen pro Zeile für KC 85/3

Es wurde ein Programm entwickelt, mit dem 80 Zeichen/Zeile für Basic realisiert werden. Dabei bewirkt das Ersetzen von PRINT durch PRINT#3 (auch mit Zusätzen wie AT, COLOR usw.) die veränderte Darstellung der entsprechenden Ausschrift auf dem Bildschirm. Weiterhin ist es möglich, über LIST#3 das Auflisten des Basic-Programms mit 80 Zeichen/Zeile zu veranlassen. Das führt zu erhöhter Übersichtlichkeit, weil keine Programmzeile mehr die Länge einer Bildschirmzeile überschreitet.

Das Zusatzprogramm liegt einschließlich der neuen Zeichenbildtabelle (Codes 20H ... 7FH) im Adreßbereich BA00H ... BC1FH. Daneben gibt es eine abgewandelte Version, die in einer REM-Zeile eines Basic-Programms untergebracht werden kann.

Interessenten steht das Programm unentgeltlich zur Verfügung.

Wolfram Schütze, Althainitz 2a, Großpostwitz, 8603

Wir suchen ...

... eine Softwarelösung für die Registratur von TGLs, Katalogen, GBls, NVs und NVes.

VEB Bau Grimmen, Produktionsbereich Berlin, Klement-Gottwald-Allee 292, Berlin, 1120; Tel. 3651300

Seidel

... eine Hard-/Softwarelösung zum Datentransfer zwischen einem ½"-Magnetband CM 5300.01 und einem PC 1715 oder AC 7100.

VEB Rationalisierung, Möwenburgstraße 13–17, Schwerin, 2756; Tel. 5651

Palinski

... dringend Software für KC 87, welche sich für Ausbildungszwecke in metallverarbeitenden Berufen eignet.

VEB Draht- und Seilwerk Rothenburg, Betriebsschule, Friedensstraße 21, Rothenburg, 4341; Tel. Könnern 7371

Brumann

Neue Medien – Totale Mattscheibe und neue Informationsordnung

von H.-J. Heinze, Urania-Verlag 1988, 160 Seiten, DDR 3,60 Mark

Mit dem Buch der Klartext-Reihe stellt sich die erste geschlossene Darstellung der Problematik der Medienentwicklung der BRD aus der Sicht der DDR-Gesellschaftswissenschaft vor. Dieser Versuch kann als geglückt bezeichnet werden!
Eingeleitet wird die Arbeit mit Zukunftsvisionen bürgerlicher Journalisten über die Folgen der stürmischen Entwicklung der Masssenmedien für den Menschen und die Gesellschaft. Im darauf folgenden Abschnitt stehen die technische Vielfalt und Reife sowie die derzeitigen Chancen der Verbreitung von Mediensystemen im Mittelpunkt. Allerdings ist hierbei anzumerken, daß der technischen Komponente kaum Aufmerksamkeit geschenkt wird. Während sich im weiteren der Verfasser mit der Verbreitung der *neuen Medien* im Verhältnis Kapital–Politik–Gesellschaft am Beispiel hochentwickelter kapitalistischer Staaten beschäftigt, behandelt der letzte Abschnitt die Perspektive der neuen Medien, ihre Bedeutung für die Durchsetzung der Menschheitsinteressen und die Notwendigkeit und Schwierigkeit einer internationalen Mediengesetzgebung.
Eine künftige Behandlung der Thematik *neue Medien* verlangt nicht nur die Darstellung der Entwicklung in den imperialistischen Staaten mit Sicht auf die speziellen Probleme der Entwicklungsländer, sondern erfordert ebenfalls Überlegungen zur Medienentwicklung in der sozialistischen Gesellschaft. Dazu ist die Frage zu klären: Wo stehen die sozialistischen Staaten bei der Schaffung eines technisch möglichen, interkontinentalen Medienverbundnetzes und wie sehen sie ihre Rolle darin?
Zusammenfassend ist festzustellen, daß die Aufgabe, die sich der Verfasser gestellt hatte, die Darstellung des Prozesses der Medienentwicklung sowie die Suche nach kritischen Bewertungen und Einschätzungen, erfüllt wurde.

H. Hoffmann

Die innovativen 80286/ 80386-Architekturen

Teil 1: Der 80286 (296 Seiten); Teil 2: Der 80386 (532 Seiten), von Klaus-Dieter Thies, te-wi Verlag GmbH, München, 1988

Mit der großen Verbreitung der AT-kompatiblen Personal Computer und neuerdings auch des Personal System/2 gewinnen Beschreibungen der Prozessoren 80286 und 80386 mehr und mehr an Bedeutung. Aus diesem Grund ist das Erscheinen der beiden Bände in deutscher Sprache unbedingt zu begrüßen.
Der Teil 1 behandelt den 80286. Er enthält zunächst eine allgemeine Einführung in die Speicherorganisation älterer Mikrorechnersysteme, erläutert dann die Speicherverwaltung einschließlich virtueller Adressierung beim 80286 sowie die zugehörigen



Schutzmechanismen und beschreibt danach die Arbeitsweise des Prozessors in Multitask-Systemen, insbesondere den dazu notwendigen Statuswechsel. In weiteren Kapiteln werden die Interruptverarbeitung, die Behandlung von Ausnahmen, Ein- und Ausgaben zur Peripherie, die Einbindung des numerischen Prozessors 80287 sowie der System Builder erläutert.
Im Teil 2 sind alle Kapitel des 1. Teiles sinngemäß enthalten, aber auf den Prozessor 80386 bezogen. Da beide Prozessoren in vielen Eigenschaften identisch sind, hat der Autor große Teile von der 80286-Beschreibung wörtlich in den 2. Teil übernommen. Dieser 2. Band enthält zusätzlich je ein Kapitel über den virtuellen 8086-Modus und über die DEBUG-Unterstützung.
Insgesamt kann festgestellt werden, daß beide Bücher im wesentlichen die Beziehung der Prozessoren zum Hauptspeicher und zur Peripherie beschreiben, also die Innovationen bzgl. Speicherverwaltung gegenüber dem 8086/8088. Unter der Architektur eines Prozessors wird in der Regel jedoch weit mehr verstanden, so daß die Erwartungen vieler Leser sicherlich nicht erfüllt werden. Sehr erschwerend für das Verständnis des Buches wirkt die fehlende Beschreibung des Befehlssatzes. Im Band 1 ist nicht einmal eine Erläuterung des internen Registersatzes beim 80286 enthalten. In beiden Büchern wird die Kenntnis des 8086/8088 vorausgesetzt. Da nun wesentliche Probleme der beiden Prozessoren nicht behandelt werden, wären Hinweise auf ergänzende Literatur dringend erforderlich. Derartige Vermerke sind aber nur in geringer Zahl enthalten. Beiden Büchern fehlen sogar Literaturverzeichnisse, obwohl viele Abschnitte von den Intel-Originalschriften (z. B. 80386 Programmer's Refererence Manual) übersetzt wurden. Trotzdem können die beiden Bücher Systemprogrammieren empfohlen werden, die den 8086/8088 bereits kennen und sich in die Innovationen von 80286 und 80386 einarbeiten wollen, da sie leicht verständlich geschrieben sind.

Dr. G. Kinnemann

Zeitschriftenumschau

G. V. Jermakov: **Testen von MIXAL-Programmen mit Hilfe symbolischer Programmausführung**

Programmirovanije 14 (1988) 1, S. 12

Die symbolische Programmausführung ist eine relativ neue Fehlernachweismethode. Ihr Wesen besteht darin, daß das Programm in einem vereinfacht gesagt algebraischen Ausdruck überführt wird, der die Transformation der Eingangsdaten widerspiegelt. Prädestiniert sind für die symbolische Ausführung vor allem Programme der numerischen Mathematik und allgemeine Programme mit einfacher Struktur der Eingangs- und Ausgangsdaten. Grundsätzliche Schwierigkeiten entstehen bei starker Verzweigung des Steuerflusses, Operationen mit Komponenten strukturierter Datentypen, Behandlung von Zyklen, Behandlung von Dateneingaben und Prozeduraufrufen. Diese Einschränkungen werden in wesentlich geringerem Maße wirksam, wenn die symbolische Programmausführung nicht als selbständiges Validierungsverfahren benutzt wird, sondern als Hilfsverfahren zur Erzeugung von Testdaten. Mit dieser Anwendung der symbolischen Ausführung von MIXAL-Programmen befaßt sich der Autor des Artikels. MIXAL ist eine von Knuth geschaffene höhere Assemblersprache. Das Ergebnis der symbolischen Programmausführung ist für jeden Programmpfad ein System von Gleichungen und/oder Ungleichungen. Besitzt dieses System eine Lösung, so ist der Pfad testbar; die Lösung enthält die Testbedingungen. Zur Bestimmung dieses Systems von Gleichungen bzw. Ungleichungen wird jeder MIXAL-Befehl durch eine Formel in postfixer Form beschrieben – die symbolische Ausführung kombiniert dann die postfixen Formeln der einzelnen Befehle.

S. V. Denisenko: **Quantitativer Schätzwert der statischen Semantikanalyse von Programmen**

Programmirovanije 14 (1988) 3, S. 81

Die Zuverlässigkeit der Programme hängt in wesentlichem Maße von den beim Programmentwurf benutzten Programmiersprachen ab. Die modernen Hochsprachen (Pascal, Modula-2, Ada) begünstigen die Verringerung der Softwarefehler. Ursache dafür ist die strenge Typisierung von Daten und Prozeduren und die Beschränkung bezüglich ihrer Sichtbarkeit. Die Zuverlässigkeit der entworfenen Programme erhöht sich auf der einen Seite durch die Verringerung der Wahrscheinlichkeit des Auftretens von Programmierfehlern und auf der anderen Seite durch die größeren Fehlernachweismöglichkeiten zur Übersetzungszeit. Letzteres, als statische Semantikanalyse bezeichnet, wird oft zusammen mit der Compilation ausgeführt, aber auch durch die einzelnen Softwarewerkzeuge (Semantikchecker). In früheren Arbeiten wurde die Fehleranzahl in Programmen bei unterschiedlicher Tiefe der Semantikanalyse experimentell bestimmt. Es zeigte sich, daß die strenge Datentypisierung und die explizite Deklaration des Sichtbarkeitsbereiches eines Moduls die Fehleranzahl in den Programmen wesentlich verringert (um das 3,12- bzw. 1,57fache). Die Programmzuverlässigkeit nach der Compilation wird durch die Anzahl der unentdeckt gebliebenen Fehler bestimmt. Einen Näherungswert für diese Anzahl, die wichtig ist für die Aufwandsbestimmung der Fehlerbeseitigung, liefert die Programmetrik, die aus statistischen Kenngrößen des Programmes gewonnen wird. In diese Metriken finden allerdings die Spracheigenschaften keinen Eingang. In dem Artikel wird der Versuch unternommen, den quantitativen Zusammenhang zwischen Programmeigenschaften und indirekt den Spracheigenschaften auf der einen Seite und der Anzahl der unentdeckten Fehler einer bestimmten Klasse – der Elementarfehler – herzustellen.

R. V. Verschigora: **Funktionskopplung mittels Koprogrammmechanismus in der Sprache C**

Mikroprocessornyje sredsta i sistemy 5 (1988) 4, S. 53

Der Koprogrammechanismus ist eine einfache und wirksame Form der informationellen Kopplung zwischen einzelnen Funktionen. Um mit der traditionellen Struktur *aufrufendes Programm – Unterprogramm (Funktion)* einen gleichen Effekt zu erzielen, sind Mehrfacheintritts- und Austrittspunkte erforderlich. Der Koprogrammechanismus erhöht die Verständlichkeit der Programme (bei adäquaten Problemstellungen). Die logischen Verknüpfungen im Algorithmus werden vereinfacht und Systemressourcen werden im geringerem Maße in Anspruch genommen. Der Autor analysiert verschiedene Varianten der nachträglichen Implementierung des Koprogrammechanismus in C:
– Modellierung in C
– Modellierung mit Assemblerfunktionen.
Zu realisierende Funktionen dabei sind:
– Initialisierung/Umwandlung der linearen Liste der Aktivierung der Funktionsaufrufe in eine baumartige Struktur
– Wiederherstellung des Zustandes der erneut aufgerufenen Funktion und Ketten des aktuellen Zustandes vor dem Aufruf
– Aufhebung der Koprogrammverbindung. Dabei geht die Steuerung an ein Koprogramm über, das im Aufrufbaum nicht unter dem aufrufenden liegt. Die unteren Zweige des Aufrufbaumes werden gelöscht.
Der Autor wählt für Koprogrammechanismus die Modellierung mit Assemblerfunktionen und gibt drei Unterprogramme PLACE, DECLINE, RESUME zur Realisierung der notwendigen Funktionen an.

zusammengestellt von Dr. B. Stiefel



Bericht

ICCD '88

Vom 3. bis 5. Oktober 1988 fand in Rye Brook (New York – USA) die International Conference on Computer Design (ICCD '88) der IEEE statt.
Die ICCD ist die führende internationale Konferenz für den Einsatz und die Anwendung der Höchstintegration in Computern und umfaßt alle Aspekte des Entwurfs und der Anwendung von VLSI-Computer- und Prozessorsystemen. Durch den multidisziplinären Charakter der Konferenz wurden die gegenseitigen Einflüsse zwischen Architektur, rechnergestütztem Entwurf, Schaltkreistest und VLSI-Basistechnologien deutlich. Die Beteiligung von etwa 500 Spezialisten zeigte deutlich ein gestiegenes Interesse an dieser Konferenz, die als einzige diesen multidisziplinären Aspekt des Computerentwurfs betont. Die internationale Beteiligung dokumentierte sich in Vorträgen aus 11 Ländern, wobei die USA mit 101 von 130 Vorträgen dominierten.
Jedoch war auch neben den fünf japanischen Beiträgen vor allem eine gegenüber früheren ICCDs gestiegene europäischen Beteiligung zu bemerken, aus der deutlich wird, daß in Westeuropa intensive Bemühungen im Gange sind, durch Beherrschung der Mikroelektronik, speziell der Höchstintegration, die entstandenen Abhängigkeiten von den USA und Japan abzubauen. Der Anteil der USA-Hochschulen und Universitäten zeigt, daß in der Zusammenarbeit zwischen Industrie und Hochschulen ein höheres Niveau erreicht wurde, daß die Hochschulen enger an die Forschungsziele der führenden Konzerne mit konkreten Aufgaben gebunden werden.
Die ICCD '88 stand unter dem Motto „Supercomputer – Herausforderung, Entwurf und Anwendung". Diesem roten Faden waren die Vorträge in vier Sektionen zugeordnet, welche die jüngsten und wichtigsten Neuerungen und die darauf aufbauenden perspektivischen Aufgaben darstellten. In der *Sektion „VLSI-Technologien"* spielte der Einfluß der ASICs auf den Computerentwurf die Hauptrolle, wobei neben dem gegenwärtigen Stand der Technik die zukünftigen Anforderungen diskutiert wurden.
Schwerpunkte waren die Anforderungen der Hersteller von Supercomputern und Super-Mini-Computern (z. B. Cray, IBM, Convex Computer Corporation) an integrierte Schaltkreise, speziell ASICs, Gehäuse in Verbindung mit Leiterplatten, Entwurfsmethoden und Entwurfswerkzeuge und Testmethoden für Schaltkreise und Systeme. In den Beiträgen wurde vorausgesetzt, daß die Entwicklung der Basistechnologien – ausgedrückt in der Erhöhung des Integrationsgrades und der Verringerung der minimalen Strukturmaße – ungebrochen weiter anhält und dabei die Speicherschaltkreise als „Technologielokomotiven" fungieren.
Hersteller von Supercomputern nutzten die Konferenz, um neue Entwicklungslinien vorzustellen und zu diskutieren, z. B. das neue C-Serien-System von Convex und das Y-MP-System von Cray.
In der *Sektion „Entwurf und Test"* wurde ein breites Spektrum wichtiger Probleme des Tests von VLSI-Chips und -Arrays, die in Rechnersystemen eingesetzt sind, zur Diskussion gebracht. Zum Thema „Design for Testability" (= testfreundlicher Entwurf) unterstrichen die Bedeutung dieses Gebietes. Sowohl Hardware- als auch Softwarelösungen müssen dazu beitragen, die Zurückweisungsquote in der Mitte der 90er Jahre auf 1 ppm (parts per million) zu senken.
In der *Sektion „Rechnergestützer Entwurf (CAD)"* wurde davon ausgegangen, daß nach Jahren einer sprunghaften Entwicklung seit Beginn der 80er Jahre gegenwärtig eine solche Reife erreicht ist, daß mit den vorhandenen Entwurfswerkzeugen der technologische Fortschritt auch sinnvoll genutzt werden kann. Schwerpunkte waren neue Simulationstechniken und Schaltungssynthese in den oberen Entwurfsebenen (Architekturniveau).
Die *Sektion „Architekturen und Algorithmen"* konzentrierte sich auf die Frage „Wann wird der breite Einsatz von Parallelcomputern zur Realität werden?" Mit der Darstellung des gegenwärtigen Standes der Architektur, des Entwurfes und der Technologie von Supercomputern, Parallel-Prozessoren und fortgeschrittenen Mikroprozessoren wird eingeschätzt, daß in den 90er Jahren der Übergang zu Parallelcomputern vollzogen wird.
Daneben wurden aber auch die gegenwärtigen Entwicklungen von Mikroprozessoren mit neuen Lösungen auf den Gebieten
– Entwurfstechnik einschließlich Befehlsreorganisation
– Pipelining auf Basis von Cache-Speichern
– Systemunterstützung für Mikroprozessoren (z. B. BUS-Entwurf, Compiler-Technologie, Cache-Zusammenarbeit)
diskutiert.
An Hochleistungscomputern mit neuen Architekturen waren u. a. interessant:
– Cydra 5, eine Maschine mit sehr langem Befehlswort (Very-long instruction word – VLIW)
– ESA/370 von IBM.
Insgesamt wurde deutlich, daß alle diese Schritte zur Erhöhung der Leistungsfähigkeit von Computern auf der Basis
– von schnellen, höchstintegrierten Speicherschaltkreisen für Haupt- und Schnellspeicher (Caches)
– von hochintegrierten, schnell und flexibel entwerfbaren ASICs und in zunehmendem Maße auch
– von einer Kombination dieser Schaltkreisklassen, den sogenannten ASMICs (application specific memory integrated circuits – Anwendungsspezifische Speicherschaltkreise)
erfolgen.
Stellvertretend für die letzte Gruppe, die ASMICs, ist ein Beitrag von TOSHIBA „Large Memory embedded ASICs" zu nennen. Es wurde eine gemeinsame Intergration eines 1-MBit-DRAMs mit einem 72-Tausend-Gatter-ASIC (See-Konzept) vorgestellt. Das Chip von 15 × 15 mm^2 wurde in einer weiterentwickelten 1-μm-Technologie entworfen.

Trendeinschätzung

Auf der ICCD '88 wurden die Fortschritte auf all den Gebieten, die mit dem Entwurf von Computern verknüpft sind, vorgestellt und die Entwicklungslinien sichtbar gemacht.
Besonders der Vormarsch von CMOS-ASIC-Schaltkreisen auch auf dem Gebiet der Computerentwicklung ist deutlich geworden. Damit ist ein Rückgang der bisher dominierenden ECL-Technik auf diesem Gebiet zu verzeichnen. Bei der Entwurfsmethodik erfolgt damit ebenfalls eine Ablösung der bisherigen Prozessorspezialentwürfe durch ASICs. Unter Ausnutzung des mit CMOS-Technologien erreichbaren hohen Integrationsgrades und der Flexibilität dieser Technologien bilden sich neue Schaltkreisklassen heraus, für die der Begriff „Heterointegration" genannt wurde.
Dies sind Schaltkreise, auf denen unterschiedliche Schaltungen auf einem Chip integriert sind, z. B. digital und analog; Speicher und Logik; Hochvolt- und Niedervoltschaltungen. Die weitere Entwicklung spezieller Integrationsparameter bei ASICs wird entsprechend Tafel 1 prognostiziert.
Auf dem Gebiet der Computerarchitektur erhalten Parallelcomputer eine immer größere Bedeutung. Aus der Laborkuriosität der 70er Jahre entwickelten sich erste reale Lösungen in den 80er Jahren, die den Mythos der Nichtmachbarkeit zerbrachen, so daß in den 90er Jahren mit einer Herstellung zu rechnen ist.

Dr. Jens Knobloch

***Tafel 1** Entwicklung spezieller Integrationsparameter bei ASICs*

	'86	'88	'90	'92	Grenzen[1]
Strukturniveau [μm]	1,5	1,0	0,8	0,6	0,2
effektive Kanallänge [μm]	0,9	0,7	0,55	0,4	0,1 ... 0,2
Versorgungsspannung [V]	5	5	5/3,3	3,3	0,1!
nutzbare Gatterzahl	50T	120T	200T	330T	?
Gatterverzögerung [us] (2 Input NAND mit Last 2)	0,7	0,5	0,3	0,24	0,02

[1] Die Begrenzungen wurden aufgrund thermodynamischer Effekte und mit Anwendung der Supraleitung abgeschätzt.

1. Fachtagung „Anwendung von 32-Bit-Rechenanlagen des SKR"

Von der Kammer der Technik, Bezirksvorstand Gera, und dem vor kurzem gegründeten Fachausschuß *Minicomputersysteme* wurde am 25. und 26. Januar 1989 in Gera mit 360 Teilnehmern die 1. Fachtagung zur „Anwendung von 32-Bit-Rechenanlagen des SKR" durchgeführt. Das Tagungsprogramm umfaßte 20 Vorträge zu Architektur, Einsatzcharakteristika, Betriebssystemen und weiterführenden Softwareprodukten für die 32-Bit-Rechenanlagen des Systems der Kleinrechentechnik (SKR). Im Mittelpunkt der Tagung stand schwerpunktmäßig die Anwendung des 32-Bit-Kleinrechners K 1840 aus der Produktion des VEB Kombinat Robotron.
Prof. Dr. Jungmann (IZ/TUD) gab eine Einführung in die Architektur der 32-Bit-Technik und des K 1840. *Prof. Dr. Schröder/Brunner* (IH Zwickau) berichteten über erste Einsatzerfahrungen mit der SM52/12 aus der ČSSR und *Dr. Galler* (FSU Jena) über erste Einsatzerfahrungen mit der Elektronika-82 aus der ČSSR.
Prof. Dr. Horn (IZ/TUD) gab eine Einführung in die Architektur des Betriebssystems SVP-1800, die von *Sachse* (LfA Berlin) durch spezielle Charakteristika aus Programmierersicht ergänzt wurden. *Liebold* (IZ/TUD) berichtete über Einsatzerfahrungen aus der Industrieerprobung des Betriebssystems MUTOS-1800.
Dr. Gollnick (LfA Berlin) gab eine Übersicht über Softwaresysteme des Leitzentrums für Anwendungsforschung (LfA) für den K 1840 wie INFOset, CADset, TOOLset, COMMset und Softwarepakete für den Systemverwalter, die dann durch weitere spezielle Vorträge des LfA ergänzt wurden.
Dr. Utke (IZ/TUD) stellte die SKRnet-Software für globale und lokale Rechnernetze vor und diskutierte Probleme des Aufbaus von lokalen Netzen mit einer Datenübertragungsrate von 10 MBit/s (ROLANET 2) sowie die Einbindung von Personalcomputern unter dem Betriebssystem DCP.
In einem abschließenden Komplex wurden Probleme der Anwendung und des Einsatzes der Programmiersprachen Lisp, Prolog, Ada, Fortran-77, Modula-2 und C diskutiert. Insbesondere seien hier die Berichte über das Programmiersystem EXPERT COMMON LISP von *Dr. Friedrich* (ZKI Berlin) und das Programmiersystem Ada von *Stein* (ZKI Berlin) hervorgehoben.
Abschließend sei erwähnt, daß das Interesse an dieser Fachtagung außerordentlich hoch war (aus Kapazitätsgründen konnten leider über 250 Teilnahmewünsche nicht berücksichtigt werden), was sich vor allem aus der Bedeutung der 32-Bit-Technik für fast alle Kombinate und Industriezweige bei der weiteren erfolgreichen Anwendung der Schlüsseltechnologien und Umsetzung der CAD/CAM-Strategie unserer Partei erklärt.

Prof. Dr. Thomas Horn



Entwicklungen und Tendenzen

Forschungen für 64-MBit-DRAM

Die Firma Fujitsu gab kürzlich auf dem International Electron Devices Meeting in San Francisco Details eines 64-MBit-DRAMs bekannt. Erste Muster sollen in den nächsten fünf Jahren vorliegen.
Die von Fujitsu angewendete Technik der *dreidimensionalen Kapazitätszellen* soll zwei wesentliche Neuerungen aufweisen. Erstens besitzen die Kapazitätszellen eine besonders feine und horizontale Struktur und zweitens wurde mit einer 0,2-μm-Technologie eine Speicherzelle von 0,9 μm² erzeugt. Diese Speicherzelle ist damit rund 20mal kleiner als die des 1-MBit-DRAMs, der in einer 1-μm-Technologie gefertigt wird. *MP*

Leiterplattenentflechten mit Transputerhilfe

Von der Firma Ziegler Instruments wurde ein transputerbestücktes Zusatzboard für PCs entwickelt, um die Arbeit mit dem Softwarepaket Caddy-Autorouter zum Entflechten komplexer Leiterplatten beschleunigen zu können. Das gesamte Autorouting wird von der CPU des PCs auf den Transputer verlagert; dieser erlaubt mit eigenem Speicher und seinen Kommunikationskanälen durch Parallel-Processing Echtzeitverarbeitung. Die Prozessoren sind mit 20 MHz getaktet und erreichen – je nach Version des Boards – eine Rechenleistung von 10 oder 20 MIPS. Damit soll es möglich sein, das Autorouting bis zu zehnmal schneller als auf herkömmlichen PCs ausführen zu können. *MP*

Der PC-Markt '88
in den USA ...

Laut einer Analyse des Marktforschungsunternehmens Store Board Inc. wurden 1988 in den USA allein über den Fachhandel mehr als 2,8 Millionen PCs verkauft. Davon waren etwa 80 Prozent mit Intel-Prozessoren ausgestattet. Der Anteil des 80286 an diesem Segment betrug 54 Prozent, des 8088/8086 18 Prozent und des 32-Bit-Prozessors 80386 18 Prozent. Marktführer sind IBM mit etwa 907000 verkauften PCs, Compaq mit etwa 455000 Einheiten und Apple mit etwa 440000 Stück. Im Business-PC-Markt nennt Store Board als Spitzenreiter IBM mit 34 Prozent US-Marktanteil (1987 41 %), gefolgt von Compaq mit 24 Prozent (1987 24%) und Apple mit 11 Prozent (1987 12%). Fast die Hälfte aller in den USA verkauften Laptops – 57000 Stück – stammten von Toshiba, so daß die Firma hierbei Marktführer bleibt.

... und in der BRD

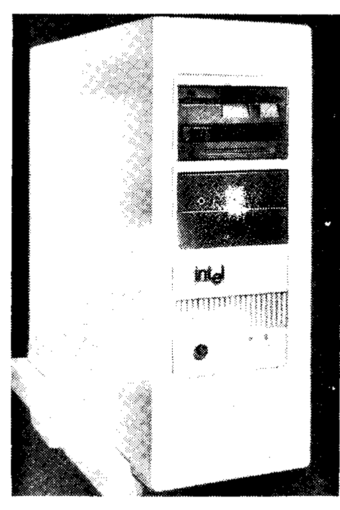
Das Marktforschungsinstitut IDC nennt für 1988 einen Absatz von 1,52 Millionen PCs plus 58000 tragbare PCs. Bei den Desktops dominieren noch 8-Bit-Prozessoren mit 30 Prozent, gefolgt vom 80286 mit 27 Prozent, Motorola 680×0 mit 24 Prozent, 8088/8086 mit 14 Prozent und dem 80386 mit 5 Prozent. 1987 waren etwa 1,4 Millionen PCs verkauft worden; der Anteil der 80386-PCs lag bei 1,3 Prozent und der der 8088/8086-PCs noch über 45 Prozent. Entsprechend dieser Entwicklung seien laut IDC 8088/8086-PCs nun in die Heimcomputerwelt gerutscht. Mit der Verfügbarkeit von OS/2 als Nachfolgebetriebssystem für MS-DOS wird 1988 als das Jahr der massiven Vorbereitung eines bevorstehenden Generationswechsels im PC-Markt bezeichnet. Bemerkenswert seien die verschwimmenden Grenzen zwischen Hochleistungs-PCs und Workstations. Während die derzeitige Leistung der Workstations in etwa einem Jahr auch von PCs erbracht werden, wird bei Workstations eine Entwicklung in Richtung Minis und Superminis erwartet. Die Marktsättigung für Workstations sei mit weniger als 10 Prozent noch lange nicht erreicht. *MP*

„Raucherfreie" Halbleiterfertigung

Laut einer Studie des US-Konsortiums Sematech, einer Forschungskooperation zur Herstellung modernster Halbleiter, atmen Raucher noch Stunden nach der letzten Zigarette soviel Schmutzpartikel aus, daß sie im Vergleich zu Nichtrauchern die zehnfache Luftverschmutzung verursachen. Mit Beginn dieses Jahres hat deshalb der amerikanische Halbleiterproduzent Advanced Micro Devices (AMD) eine Kampagne gestartet, um Raucher aus den Reinräumen in seinen beiden texanischen Werken zu verbannen. Allerdings erhalten die Raucher zunächst Gelegenheit, sich in kostenlosen Kursen das Rauchen abzugewöhnen. Sollten die Ergebnisse der Kampagne positiv sein, droht auch den Rauchern in den kalifornischen Werken ein Verbot zum Betreten der Reinräume. *MP*

Intel System 520

Als bedeutender Schaltkreishersteller ist die Firma Intel allgemein bekannt. Weniger bekannt dürfte sein, daß sie – laut Infocorp und Dataquest – bei sogenannten Midrange-Computern nach IBM und DEC sowie vor Hewlett-Packard bei verkauften Systemen den dritten Platz einnimmt.



Als erstes Produkt mit der neuen *Multibus II Systems Architecture* (MSA) und auf der Basis des 80386 wurde vor kurzem das System 520 präsentiert. MSA besteht aus einer Hierarchie von Hardware, Firmware sowie Software-Schnittstellen und Protokollen und erweitert den Multibus II in Richtung Systementwurf, da sie die Zusammenarbeit auf der Systemebene fördert. Die tragenden Pfeiler der Architektur sind: skalierbares Multiprocessing, das Multibus II-Transportprotokoll und Message Passing, Multiprozessor-Systemboot und -Initialisierung sowie System- und Board-Diagnose. Das System 520 läßt sich auf bis zu vier CPU-Platinen ausbauen, läuft unter dem Echtzeit-Betriebssystem iRMX II und verfügt über einen SCSI-Peripherie-Controller sowie einen E/A-Server mit Grafik in Fenstertechnik. Das System nutzt die Multibus II-Busplatine als schnelles Netz, das den voneinander unabhängigen Prozessoren das Arbeiten als vernetzte Systeme erlaubt. Jeder Prozessor kann mit jedem anderen Prozessor über die Busplatine und mit weiteren vernetzten Systemen File-Sharing und File-Transfer betreiben sowie als virtuelles Terminal dienen. Das System gibt es in drei Konfigurationen.
Das OEM-Basissystem im Standgehäuse (daneben ist auch ein Tischgehäuse verfügbar) mit acht Steckplätzen enthält den SCSI-Controller iSBC 386/258, das zentrale Service-Modul, eine 540-Watt-Stromversorgung, Floppy-Disk-Controller und -Laufwerk sowie die Dokumentation.
Als „base plus" kommen dazu noch eine 20-MHz-CPU-Karte mit 4 MByte DRAM, ein Grafik-Controller, ein Sechskanal-Terminal-Controller, ein Ethernet-Controller, 380-MByte-Festplatte und ein 150-MByte-Bandlaufwerk.
Interaktive Grafik mit schneller Fenstertechnik bietet das erstmals im System 520 eingesetzte Grafik-Subsystem iSBX 279. Es eignet sich für Platinen und Systeme, die unter iRMX laufen. Es wird über einen Standard-SBX-Steckverbinder an die Platinen angeschlossen. Das Subsystem verwendet den Display-Prozessor 82786 für die nahezu verzögerungsfreien Fenstermanipulationen. Es wird mit einer Bibliothek von Applikationsschnittstellen geliefert, die einen umfangreichen Satz von Prozeduraufrufen zur Steuerung der Bit-map-Grafik und der Fenster enthält. Zusätzliche menügestützte Software ermöglicht die interaktive Manipulation mit einer Maus. Mehrere einander überlappende Fenster mit grafischen oder alphanumerischen Informationen lassen sich darstellen. *MP*

Flache Kathodenstrahlbildröhre

Nach dem Prinzip des Strahlindex hat die Firma Matsushita eine flache Kathodenstrahlröhre entwickelt, die in ihren Eigenschaften und in bezug auf die Geometrie den handelsüblichen Farbbildröhren entspricht.
Die Bildröhre soll eine Länge von 31,7 cm, einen Halsdurchmesser von 20 mm und eine Diagonale von 15,2 cm aufweisen. Der Hals der Bildröhre zeigt senkrecht nach unten. Durch diese konstruktive Lösung beträgt die Einbautiefe nur 77 mm. Matsushita hat mehrere Jahre benötigt, um die Nachteile des schon lange bekannten Strahlindexverfahrens zu beseitigen. Dabei mußten Lösungen für die schwer beherrschbaren Helligkeitsunterschiede zwischen dem oberen und dem unteren Bildteil gefunden werden. Die Indexröhre arbeitet im Gegensatz zu normalen Bildröhren statt mit drei nur mit einem Elektronenstrahl für die Farben rot, grün und blau. Die Farben werden durch rote, grüne und blaue vertikale Farbstreifen auf der Frontplatte beim Überstreichen mit dem scharf gebündelten, energiereichen Kathodenstrahl erzeugt. Die Steuerung der Farben erfolgt durch zusätzliche Fotodioden und einen Fotoverstärker.
Indexröhren arbeiten nicht mit einer Schattenmaske. Dadurch treten keine Farbverfälschungen durch Aufheizen der Maske und durch erdmagnetische Einflüsse auf.
Quelle: Blick durch die Wirtschaft vom 10. 01. 1989 *Wi*

Bleistifte bleiben Arbeitsmittel

Kugelschreiber, Tintenstifte und Tuschespitzen sind die gebräuchlichsten Zeichengeräte bei Plottern. Für den Konstrukteur oder Architekten ist jedoch der Bleistift das traditionelle Arbeitsmittel.
Die Firma CalComp setzt den Bleistift ein, um preiswerte Plotterausgaben mit höchster Geschwindigkeit zu realisieren. Diese Ausgaben gestatten es, Änderungen durch Radieren und manuelles Ergänzen vorzunehmen. Nach der optischen Prüfung der Zeichnung kann die Reinzeichnung mit der erforderlichen Genauigkeit und Qualität in Tusche erfolgen.
Ein manueller Wechsel der Stifte ist nicht erforderlich, da sich alle Stiftarten in einem Karussell befinden, daß entsprechend angesteuert werden kann.
Quelle: BZB-Sachmagazin. – Hamburg 92(1989)1. – S. 38 *Wi*

100 MByte auf Glas-Festplatte

Auf der Suche nach immer neuen Möglichkeiten, die Speicherkapazität zu erhöhen und neue Substrate zu verwenden, gewinnt der Rohstoff Glas mehr und mehr an Bedeutung.
Die kalifornische Firma Areal Technology Corp. konstruierte eine neuartige Festplatte, die als Basis für die aufgesprühte Magnetschicht ein Glassubstrat verwendet. Die Festplatte verfügt über eine Speicherkapazität von 100 MByte und befindet sich in einem $2{,}5 \times 10 \times 13\,cm^3$ großen Gehäuse.
Die Schreibdichte wird mit 57000 Bit pro Zoll in der inneren Spur angegeben. Durch eine Verkleinerung der Dünnschicht-Schreib-/Leseköpfe konnte der Abstand zwischen Plattenoberfläche und Kopf auf 4 μm verkürzt werden.
Das zunächst als Prototyp unter der Bezeichnung BP-100 fertiggestellte Festplatten-Laufwerk ist für Anwendungen in Laptop-Computern vorgesehen.
Quelle: Elektronik. – München 38 (1989) 1. – S. 7 *Fa.*

EAW electronic P 8000 compact

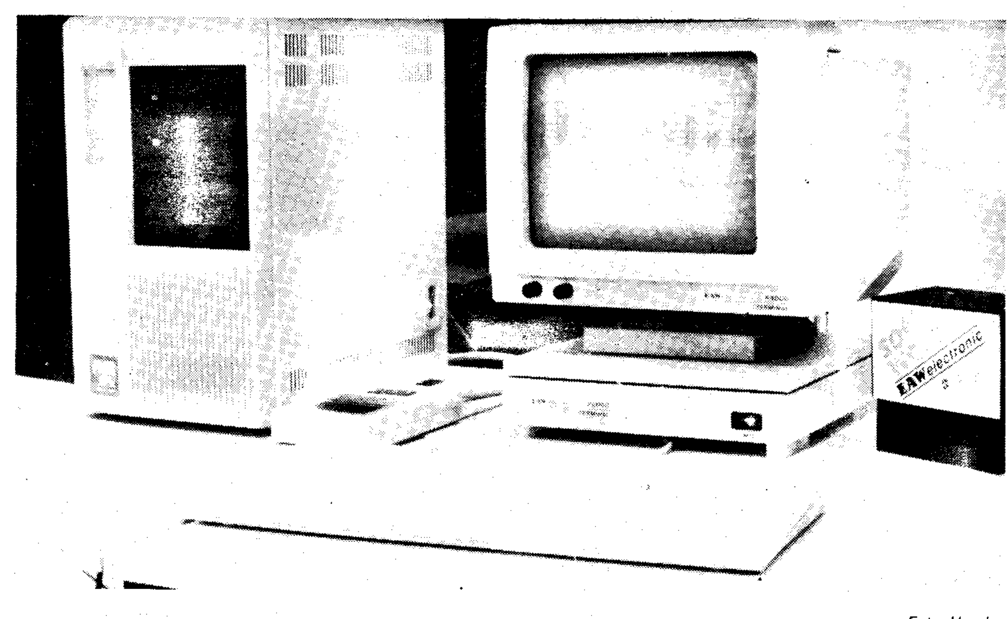


Foto: Hemke

Auf der Leipziger Frühjahrsmesse 1989 präsentierte das Kombinat Elektro-Apparate-Werke Berlin-Treptow das Programmier- und Entwicklungssystem P 8000 compact. Der P 8000 compact ist der erste Rechner mit dem ebenfalls erstmals auf der LFM '89 ausgestellten schnellen 16-Bit-Mikroprozessorsystem U 80600 vom Kombinat Mikroelektronik Erfurt (siehe auch Seite 130).

Mit dem P 8000 compact setzt das Kombinat EAW eine Produktlinie fort, die 1984 mit dem GDS 6000 (A 5120-kompatibel, Betriebssystem UDOS) begann. Nachdem der GDS 6000 1985 um den Prozessor U 8000 erweitert wurde, entwickelte das EAW-Forscherteam 1987 den P 8000 (Prozessorsysteme U 880 und U 8000, Betriebssysteme WEGA, UDOS, OS/M und IS/M). Der P 8000 compact, der 1989 den P 8000 ablösen wird (vergleiche MP 3/1987, Seite 68), ermöglicht mit dem (dritten) Prozessorsystem U 80600 zusätzlich die Nutzung des Betriebssystems WDOS. Durch die Integration der Winchesterlaufwerke in das Grundgerät (beim P 8000 war dafür ein Beistellgerät erforderlich) wurde beim P 8000 compact eine Volumenreduzierung von 50 % erreicht.

Während die zwei Harddisklaufwerke nur von den Betriebssystemen WEGA und WDOS verwaltet werden, dienen die zwei Floppy-Disks im Grundgerät den Betriebssystemen UDOS und OS/M als Massenspeicher (unter WEGA und WDOS werden sie nur für die Backup-Speicherung verwendet). Für UDOS und OS/M können optional zwei weitere externe Floppylaufwerke angeschlossen werden.

Auf der **U 8001**-CPU-Karte befinden sich fünf Steckplätze (Slots). Ein Steckplatz wird von der batteriegepufferten Systemuhr belegt. Die anderen vier Steckplätze können bis zu vier 1-MByte-RAM-Karten (mit 256-KBit-Chips) für den U 8001 aufnehmen (adressierbar sind 8 MByte). Damit stehen dem Betriebssystem WEGA 4 MByte RAM zur Verfügung. Für den Einsatz des Betriebssystems WDOS werden zwei Steckplätze, für die **U 80601**-CPU-Karte und eine 1-MByte-Dualport-RAM-Karte, benötigt. Das geht zu Lasten des WEGA Hauptspeichers, so daß WEGA dann noch 2 MByte RAM verwalten kann und 1 MByte RAM von WDOS und WEGA parallel verwaltet wird.

Der P 8000 compact ist mit acht V.24-/IFSS-Kanälen ausgestattet. An diese Kanäle können wahlweise P 8000-Terminals, Drucker, Emulatoren oder *Remote*-Computer angeschlossen werden. Die *Remote*-Computer sind PCs, die unter WEGA als Terminals im Multiuserbetrieb arbeiten.

Der EPROM-Programmer wird über eine parallele 8-Bit-Schnittstelle angeschlossen und ist für die Programmierung der EPROM-Typen 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A und 27512 geeignet.

Für den P 8000 compact werden dem Nutzer vier Betriebssysteme zur Verfügung gestellt. Dem Multiuser- und Multitaskingbetrieb dient das UNIX-kompatible **WEGA** (UNIX-Version 7 bzw. UNIX-System III). Es läuft auf dem Prozessor U 8001 und enthält ein hierarchisches Dateiverwaltungssystem, die Möglichkeit der Ein-/Ausgabeumlenkung, Möglichkeiten zur Pipe- und Filterverarbeitung sowie einen Shell-Kommandointerpreter. Die Systemsprache ist C. Zum WEGA-System gehören rund 200 Dienstprogramme, die den *UNIX-Werkzeugsatz* bilden, weiterhin Assembler und C-Compiler, für den U 8000, Programme für Textverarbeitung (nroff, troff) und Rechnerkopplung (uucp, remote) sowie ein System für die Verwaltung von Anwenderquellprogrammen (**s**ource **c**ode **c**ontrol **s**ystem). (Siehe auch MP 8/1988, Seite 227.)

Das zweite 16-Bit-Betriebssystem ist das MS-DOS-teilkompatible **WDOS**. Es läuft auf dem Prozessor U 80601 und kann an einem der maximal acht Terminals unter WEGA-Steuerung im Singleuserbetrieb genutzt werden. Dem Anwender stehen 640 KByte RAM zur Verfügung. Es können System- und Anwenderprogramme von MS-DOS-kompatiblen Rechnern abgearbeitet werden, sofern sie zeilenorientiert arbeiten. WDOS kann parallel zu WEGA arbeiten, da es auf einem zweiten Prozessor läuft.

Die beiden 8-Bit-Betriebssysteme sind das RIO-kompatible **UDOS** und das CP/M-kompatible **OS/M** (Version 2.2), die auf dem Prozessor U 880 abgearbeitet werden. Während UDOS Softwaresysteme für die Entwicklung von Anwendersoftware der Prozessoren U 880, U 881 ... 886 und U 8000 enthält, kann mit OS/M die Effektivität der Hilfs- und Nebenprozesse, insbesondere Textverarbeitung (WS-kompatible) und Datenbankarbeit (dBase II-kompatibel) erhöht werden.

Mit dem P 8000 compact wird den Hard- und Softwareentwicklern ein System in die Hand gegeben, das die Programmentwicklung und -testung für alle in der DDR verfügbaren Mikroprozessortypen (U 880, U 881 ... 886, U 8000, K 1810 WM86/88, U 80600) ermöglicht. Dabei kann im Multiuserbetrieb gearbeitet werden.

MP-Hk

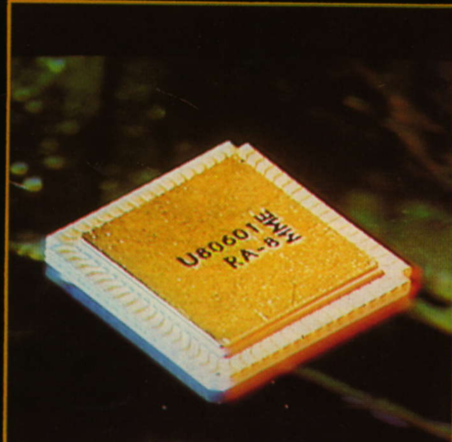
Technische Parameter

Abmessungen des Grundgerätes (HxBxT)	420 × 260 × 395 mm³
Interface (seriell)	V.24 (maximal 10 m) IFSS (maximal 500 m)
Floppy Disks	2 × 5¹/₄" mit maximal je 1,6 MByte optional 2 × 5¹/₄" extern
Harddisks Steckplätze Hauptspeicher U 8001-CPU-Karte	1 bis 2 × 5¹/₄" mit je 43 MByte 5 maximal 4 MByte DRAM auf 4 Leiterplatten zu je 1 MByte 4 serielle Schnittstellen
U 880-CPU-Karte	64 KByte DRAM 2 externe Floppies möglich
U 80601-CPU-Karte mit 1-MByte-Dual-Port-RAM-Karte	auf 2 Steckplätzen (für U 8001-CPU-Karte, dadurch noch maximal 2 MByte DRAM verfügbar)
Uhr	batteriegestützt (belegt einen Steckplatz)
Terminal	VT100-kompatibel mit Monitor K 7229 (monochrom, alphanumerisch)
EPROM-Programmer	für 2716 bis 27512
Einchiprechner-Emulator	für EMR der U 881-Reihe
Betriebssysteme	UDOS (RIO-kompatibel) OS/M (CP/M-kompatibel) WEGA (UNIX-kompatibel) WDOS (MS-DOS-teilkompatibel)
Erweiterungssoftware	WEGA-CROSS (Assembler und Compiler) für U 880, U 881, K 1810 WM86) WEGA-WORD (Textverarbeitung) WEGA-CALC (Tabellenkalkulation) WEGA-DATA (Datenbanksystem) WEGA-REMOTE (für sternförmige Rechnernetze mit PCs) IRTS-8000 (für Echtzeitanwendungen)
Programmiersprachen	WEGA-BASIC WEGA-PASCAL WEGA-FORTRAN 77
Stromversorgung	220 V $\pm^{10\,\%}_{15\,\%}$ 50 Hz, 150 VA

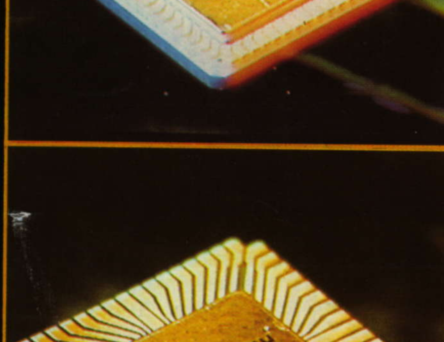
ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 5
S. 129–160

Schaltkreise des Systems U 80600

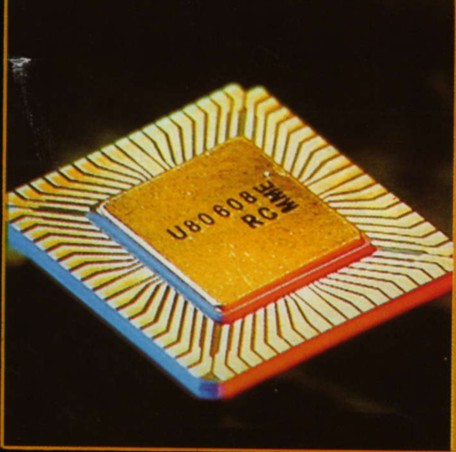
CPU U 80601



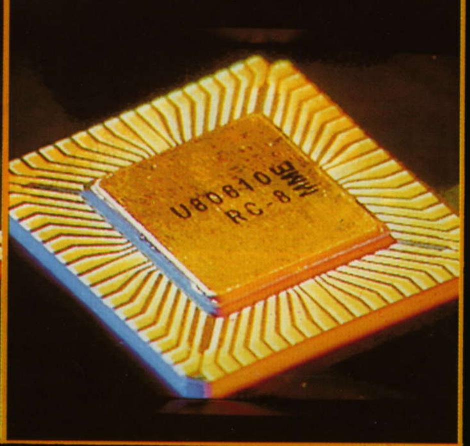
Buscontroller U 80606



EDC U 80608



DRAM-Controller U 80610



Fotos: Eberhard Mai

CPU U 80601

Gegenüber dem K 1810 WM86 hat der U 80601 einen erhöhten Befehlsdurchsatz, eine Unterstützung von virtuellen Adressierungskonzepten, einen verbesserten Zugriffsschutz und die Fähigkeit des Multitasking. Bei gleicher Taktfrequenz erreicht er die 2,5fache Rechnerleistung. Seine maximale Taktfrequenz beträgt 16 MHz. Programme, die für den K 1810 WM86 geschrieben wurden, laufen auch auf dem U 80601.

Buscontroller U 80606

Der Buscontroller hat die Aufgabe, die Statussignale der CPU U 80601 zu dekodieren und daraus die entsprechenden Lese- und Schreibkommandos für den Bus zu erzeugen. Weiterhin steuert der U 80606 die Adreßlatches und die Datenbustreiber durch entsprechende Steuersignale.

EDC U 80608

Der Schaltkreis zur Fehlererkennung und -korrektur (Error Detection and Correction) analysiert Datenverluste, die durch äußere Einflüsse wie Störstrahlung entstanden sind (Soft Errors). Er erzeugt einen fehlerkorrigierenden Code und nimmt im Fehlerfall die Korrektur vor. Der U 80608 ist für die direkte Zusammenarbeit mit dem U 80610 konzipiert.

DRAM-Controller U 80610

Der U 80610 unterstützt den Anschluß von dynamischen RAM-Schaltkreisen mit Speicherkapazitäten von 16, 64 und 256 KBit. Er kann einen Adreßraum von 2 MByte verwalten und eignet sich für den Aufbau von Dual-Port-RAMs. Zusammen mit dem U 80608 ermöglicht er die Erkennung und Korrektur von Fehlern sowie den einfachen Aufbau von großen Speicherarrays.

Lesen Sie die entsprechenden Beiträge zu diesen Schaltkreisen sowie zum SCC U 82530 und zum CIO U 82536 in diesem Heft.


ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 6 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
Filesharing
Eine Datei –
Zugriff für viele
Turbo-Pascal-Kurs

Technik international

Komfortable Drucker

Das japanische Elektronikunternehmen Star Micronics nahm die CeBIT'89 zum Anlaß, neue, auch für den Büro- und Industrieeinsatz geeignete Drucker vorzustellen.

Den Anforderungen der industriellen Anwendung an robuste Bauart, flexible Anschlußmöglichkeit sowie hohe Druckgeschwindigkeit wurde durch Geräte der Typenreihe FR (FR–10, FR–15) entsprochen. Die (theoretische) Leistung von maximal 300 Zeichen pro Sekunde in 9 möglichen Schriftarten und ein interner Speicher von 31 KByte ermöglichen im 9-Nadeldruck insbesondere den schnellen Ausdruck von Programmlistings, Lagerbestandsübersichten und Meßprotokollen, ohne den Computer während des Druckes zu blockieren.

Für den Einsatz in Büros dagegen wurden die 24-Nadel-Matrix-Drucker XB24-10 und XB24-15 (10 Zoll bzw. 15 Zoll/A4 bzw. A3) mit Fontkarte vorgestellt (Bild 1 und 2).

Eine von Star als *Super Letter Quality (SLQ)* bezeichnete Druckqualität bietet durch zweimaliges Drucken jeder Zeile mit einer Auflösung von 48 × 35 Punkten das Ergebnis eines 48-Nadel-Druckers (vgl. auch MP 6/88, S. 171) und damit etwa die Qualität von Laser Druckern (Near Laser Quality).

Ein weiterer entscheidender Vorzug der Geräte der XB-Reihe für den Büroeinsatz ist ein äußerst niedriger Geräuschpegel von 54 dB(A), der durch den Quiet-Modus noch auf 51 dB(A) gesenkt werden kann. Druckgeschwindigkeiten von 240 cps (Zeichen pro Sekunde) im Entwurfsmodus (Draft), 80 cps in Letter Quality (LQ) und 40 cps in Super Letter Quality (SLQ) drücken die Leistung der Drucker XB24–10/15 aus. Zum Standard gehören bei beiden Typen 13 Schreibmaschinenschriften, zwei OCR-Schriften und eine Barcode-Schrift, die bequem über das Bedienfeld gewählt werden können (Bild 3). Shadowed und Outlined sind in allen Schriftarten möglich. Der Schubtraktor für Endlospapier und die Friktionswalze zum Einzelblatteinzug sowie eine Papierpark-Einrichtung sind wie bei den bisherigen Star-Druckern bzw. vergleichbaren im Lieferumfang der Standardausstattung.

Als Neuerung stellt Star Micronics Erweiterungssteckkarten in der Größe einer Scheckkarte vor. Diese **FONTCARDs** können zusätzlich sowohl seltene Schriftarten als auch Zeichensatzkonvertierungstabellen enthalten, die – in den Drucker geladen – eine sehr hohe Emulationsfähigkeit der XB-Drucker gewährleisten.

Alle hier vorgestellten Drucker der FR- und XB-Reihe sind mit Farbbändern desselben Typs ausgestattet, wodurch der Einsatz mehrerer Drucker noch ökonomischer wird.

Die Lebensdauer eines Farbbandes wird mit mehr als 5 Millionen Druckzeichen angegeben. *Siehe auch 3. Umschlagseite*



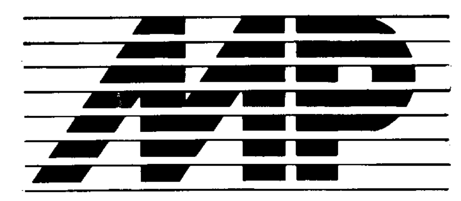




Technische Daten

	FR–10/15	*XB24–10/15*
Druckmethode	9-Nadel-Matrix	24-Nadel-Matrix
Druckgeschwindigkeit (Zeichen/Sekunde)	300 in Draft Quality	240 in Draft Quality 80 in Letter Quality 40 in Super Letter Quality (Near Laser Quality)
Schriftarten	9	16
Papierformate	A4 bzw. A3	A4 bzw. A3
Besondere Merkmale	31 KByte interner Speicher	QUIET-Modus Laden zusätzlicher Schriftarten von FONTCARDs

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 6 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelgrafik H. J. Eggstein

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

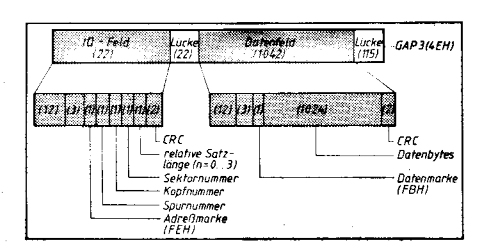
Redaktionsschluß 11. April 1989

AN (EDV) 49837

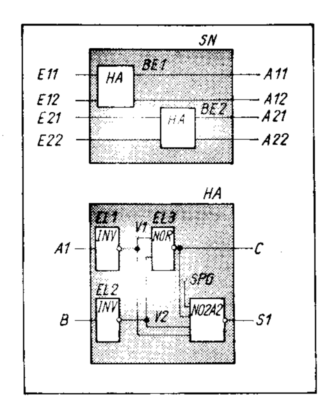
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

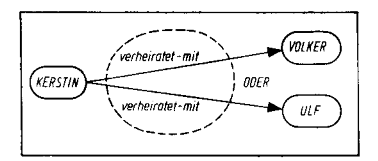
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Auf der **Seite 166** beginnen wir einen Artikel zum Floppy-Disk-Controller U 8272. Im Nicht-DMA-Modus sowie im reinen Pollingverfahren kann der Aufwand für den Floppy-Disk-Controller sowie für den Floppy-Treiber reduziert werden.



Der zweite Teil unserer ASIC-Reihe „Entwurf von Gate-Array-Schaltkreisen" auf der **Seite 168** befaßt sich mit dem Entwurfsvorgang am Beispiel des Entwurfssystems ARCHIMEDES. Für die Nutzung der 16-Bit-PC-Technik stehen die Systeme PC-GAD und MELGET zur Verfügung.



Den ersten Teil des Beitrages „Mittel und Methoden der Künstlichen Intelligenz" finden Sie auf der **Seite 179**. Er beleuchtet einige Modelle wissensverarbeitender Systeme.

Inhalt

Vorschau

Für MP 7/1989 bereiten wir für Sie unter anderem Beiträge zu folgenden Themen vor:

- Logischer Entwurf von Datenbanken
- Logikanalyse des Mikroprozessors 8086/88
- Desktop Publishing
- Leipziger Frühjahrsmesse 1989

Elektronik-Spezialisten unterstützen Betriebe

Gefragte Ratgeber waren auch im vergangenen Jahr die 45 Mitarbeiter im Berliner Ingenieurbetrieb für die Anwendung der Mikroelektronik. Mit ihrem umfangreichen Wissen lösten die Spezialisten Fragen beim Einsatz der Mikroelektronik für 20 vorrangig kleinere Betriebe, darunter 14 aus der Hauptstadt. Was dadurch für die Anwender herausspringt, beweisen 10500 eingesparte Arbeitsstunden. Neun Werktätige konnten neue Aufgaben übernehmen. Vor allem durch neue Software wurden Materialien und Energieträger für rund 1,3 Millionen Mark weniger benötigt. Darüber hinaus gaben die Fachleute den Anwendern Empfehlungen für den effektiven Einsatz eines breiten Sortiments elektronischer Bauelemente aus DDR-Produktion und RGW-Aufkommen, insbesondere zur vorbeugenden Instandhaltung von importierten Geräten und Anlagen.
Neben der Themenlösung beschäftigt sich ein Teil der Mitarbeiter mit Kundenberatung. Im vergangenen Jahr waren umfängliche Auskünfte, darunter zum Bauelementesortiment, etwa 120mal gefragt. Die Fachleute führen zusätzlich Lehrgänge für den Umgang mit moderner Technik durch. Diese Qualifikation nutzen jährlich rund 450 Werktätige in Berlin. *ADN*

Datenbanksystem für Nutzergemeinschaften vorgestellt

Das Datenbanksystem AIDOS zur Information und Dokumentation, vorgesehen vor allem für die Literaturrecherche, steht ab 1990 für mehrere Nutzer gleichzeitig zur Verfügung. Republikweit ist dann über das Datennetz der Deutschen Post parallel ein bildschirmgestützter Dialog mit der Datenbank möglich. Die neue Lösung wurde Ende Januar auf einer gemeinsamen Konferenz des Leipziger Datenverarbeitungszentrums (DVZ), des VEB Robotron-Projekt Dresden und des Zentralinstitutes für Information und Dokumentation in der Messestadt vorgestellt. Mehr als 100 Vertreter von Betrieben und Einrichtungen nahmen an der Veranstaltung teil, die Vorführungen einschloß. Als technische Basis des Recherchesystems werden ESER-Rechner EC 1057 mit großer Speicherkapazität eingesetzt. Insbesondere Anwendern mit kleinerem Auftragsvolumen kommt eine solche Datenbank entgegen. Vereint in Nutzergemeinschaften, können sie schnell und kostengünstig Informationen, so von Fachliteratur verschiedener Wissenschaftsbereiche, in die Datenbank einspeisen oder auf den eigenen Terminals abrufen. *ADN*

SPS 7000 in Serie

Eine neue speicherprogrammierbare Steuerung vom Typ SPS 7000 hat der VEB Numerik „Karl Marx" in die Serienproduktion übergeleitet. Es handelt sich um das erste Erzeugnis des modernen mikroelektronischen Steuerungssystems 7000, mit dem dieser Betrieb des Werkzeugmaschinenkombinates „Fritz Heckert" in den nächsten Jahren einen entscheidenden Beitrag zur flexiblen Automatisierung in der Volkswirtschaft der DDR leisten wird.

Die ersten Steuerungen SPS 7000 sind für den Einsatz in leistungsfähigen Bearbeitungszentren des Stammbetriebes des Fritz-Heckert-Kombinates bestimmt. Sie werden in diesem Jahr an rund 60 weitere Kunden geliefert. Die Steuerungen basieren auf der 16-Bit-Mikroprozessortechnik. Die SPS 7000 kann in Abhängigkeit vom konkreten Anwendungsfall in angepaßten Ausbaustufen geliefert werden und erhält dadurch eine große Einsatzbreite. Der VEB Numerik stellt für seine neue Steuerung auch die entsprechende Programmiertechnik bereit. Sie ermöglicht es dem Anwender, auf der Grundlage moderner, bedienerfreundlicher Programmiersprachen seine konkreten Steuerungsaufgaben zu lösen. Geschaffen wurde die Programmiertechnik von einem Jugendforscherkollektiv des Betriebes in enger Zusammenarbeit mit der Technischen Universität Dresden. *ADN*

Jahrestagung über Zusammenarbeit zwischen Honeywell und DDR-Betrieben

Im Rahmen der bestehenden langfristigen Vereinbarung über die Zusammenarbeit zwischen der Firma Honeywell Inc. (USA) und Außenhandelsbetrieben sowie Kombinaten der DDR fand Anfang Februar in Berlin im Internationalen Handelszentrum die Jahrestagung 1989 statt.
Die beteiligten Firmen und Betriebe konnten eine positive Bilanz ihrer wirtschaftlichen und wissenschaftlich-technischen Beziehungen feststellen. Honeywell errichtete beispielsweise in Zusammenarbeit mit DDR-Kombinaten, unter anderem dem VEB PCK Schwedt, Automatisierungssysteme. Betriebe der DDR lieferten Geräte und Ausrüstungen an Honeywell. Im Ergebnis der Tagung ist ein weiteres Rahmenabkommen zwischen Honeywell und Robotron abgeschlossen worden. Es sieht den Einsatz moderner Rechentechnik in Anlagen der DDR und auf Drittmärkten vor. *ADN*

Robotron/Rank Xerox-Demonstrationszentrum

Am 20. März 1989 wurde in Anwesenheit des Britischen Botschafters, Mr. Nigel Bromfield, des Generaldirektors des AHB Robotron, Herrn Dr. Abicht, und von Mr. Ralph R. Land, OBE, Generaldirektor, Rank Xerox Limited, Abt. Eastern Export Operations, London, ein Demonstrationszentrum eröffnet. Es steht seit 1. April 1989 dienstags bis donnerstags von 10 bis 15 Uhr für interessierte Kunden aus der DDR zu Demonstrations- und Informationszwecken zur Verfügung (kein Kopierservice): Robotron Export-Import, Allee der Kosmonauten 24, Berlin, 1140; Tel. 5400130.
Zur Zeit stehen dort die RX-Kopiergeräte 1012, 1025Z und 1065. Informationen über die weiterhin in der DDR vertriebene RX-Technik sind im Zentrum ebenfalls erhältlich. *Niendorf*

Mikromechanik als Partner der Mikroelektronik

Mikromechanik ist eine wissenschaftlich-technische Arbeitsrichtung, mit der das Ziel verfolgt wird, bekannte Technologien der Mikroelektronik wie die Lithographie zur Strukturierung von Halbleitermaterialien für die Herstellung mechanischer Funktionselemente und Baugruppen zu nutzen. Das heißt, es geht um die massenweise Fertigung mechanischer Elemente kleinster Abmessungen und hoher Maßhaltigkeit. Mit der Mikromechanik wird die Vereinigung von Elementen der konventionellen Mechanik mit der Mikroelektronik auf einem Chip möglich. So werden beispielsweise Impulse von Schaltkreisen über mechanische Elemente wie Sensoren und Aktoren umgesetzt. Es liegt in der Natur der Sache, daß konventionelle Mechanik mit ihrer Größe dafür nicht in Frage kommt, die Ausmaße vielmehr erheblich verkleinert werden müssen. Die Verwendung von Technologien und Grundmaterialien der Mikroelektronik hilft hier weiter. In Verbindung mit der Mikrooptik und -akustik entstehen so gerätetechnische Lösungen, die sich durch enorm kleines Volumen, reduzierten Energieverbrauch, hohe Leistung und Zuverlässigkeit auszeichnen. Zu den perspektivischen Anwendungsgebieten der Mikromechanik gehören Sensoren für die unterschiedlichsten physikalischen Größen in der Prozeßautomatisierung. Außerdem zeichnet sich ihr Einsatz in Antriebselementen, beispielsweise für hochpräzise Kleinstmotoren ab. Denkbar sind ebenso Mikrostell- und Dosiereinrichtungen in der Medizintechnik, wo ähnlich wie bei Herzschrittmachern technische Hilfen kleinster Abmessungen auch an anderen Stellen des Körpers implantiert werden können.
Aufbauend auf den in der DDR beherrschten Halbleiterbasistechnologien wird seit mehreren Jahren an der Entwicklung und Anwendung solcher mikromechanischer Grundstrukturen gearbeitet. Die Forschungen zur Mikromechanik, so zu den erforderlichen Technologien, Entwurfsregeln sowie Grundelementen der Mikromechanik, sind insbesondere an der Technischen Universität Karl-Marx-Stadt konzentriert. Im Zusammenwirken mit dem Zentralinstitut für Kernforschung Rossendorf und dem Institut für Mechanik der AdW, der Technischen Universität Dresden und der Technischen Hochschule Ilmenau wurden erste wichtige verfahrenstechnische Lösungen zum Einbringen von mechanischen Strukturen in Silizium erzielt, vor allem für die Meß- und Sensortechnik.
Für die industrielle Fertigung genutzte Forschungsergebnisse ermöglichen bereits, im VEB Geräte- und Reglerwerke „Wilhelm Pieck" Teltow die Halbleiter-Umformerreihe „audapas" herzustellen. Durch die Biegung einer wenige Mikrometer dünnen Siliziummembran mit integrierten Widerständen wird der Druck in elektrische Signale umgewandelt. Gegenüber bisheriger Druckmeßtechnik verringert sich der Materialeinsatz auf 25 Prozent. Für diese neuartigen Halbleiterdrucksensoren hoher Empfindlichkeit erhielt das Teltower Entwicklungs- und Überleitungskollektiv 1988 den Nationalpreis für Wissenschaft und Technik.

Prof. Dr. G. Montag in „Presseinformationen" Nr. 150/88

Schaltkreiszentrum des Schwermaschinenbaus

Mit der Entwicklung und dem Entwurf anwenderspezifischer Schaltkreise (ASICs) profiliert sich der Magdeburger VEB Forschung, Entwicklung und Rationalisierung des Schwermaschinen- und Anlagenbaus (FER) in seinem Industriezweig immer mehr zu einem Zentrum für die Erzeugnisinnovation auf modernster mikroelektronischer Basis. Im Schwermaschinen- und Anlagenbau wird ein zunehmend großer Teil der neuen Produkte funktionsbestimmend mit Mikroelektronik ausgerüstet. Die Ingenieure im FER entwerfen dafür in diesem Jahr zwölf Gate-Array-Schaltkreise. Das sind mehr als in den vergangenen beiden Jahren zusammen. Partner für bereits fertiggestellte Lösungen waren bisher das eigene Schwermaschinenbaukombinat „Ernst Thälmann", das Kombinat ILKA und das Schwermaschinenbaukombinat „Karl Liebknecht". Jetzt sind auch Schaltkreisentwürfe für den Aufbau lokaler Rechnernetze und für die Überwachung von Armaturen in Arbeit. *ADN*

Teures Hacken

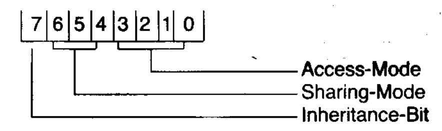
Wie die PC-Woche in ihrer Ausgabe 10/89 berichtet, wurde in den USA zum erstenmal ein Hacker anhand des *Gesetzes zum Computermißbrauch* zur Verantwortung gezogen. Der zum Zeitpunkt seiner Aktivitäten 16- bzw. 17jährige Zinn hatte unerlaubterweise mit Computern der Firma AT&T und des amerikanischen Verteidigungsministeriums kommuniziert und dadurch „wissentlich und in betrügerischer Absicht" Programme im Wert von 1,2 Millionen Dollar gestohlen und Dateien im Wert von 174000 Dollar zerstört. Außerdem veröffentlichte er Paßwörter, Telefonnummern und Tips zum Einbruch in das Sicherheitssystem von AT&T. Zinn wurde in allen Punkten der Anlage schuldig gesprochen. Die Quittung: neun Monate Freiheitsentzug und 10000 Dollar Geldstrafe. *MP*

Turbo-Pascal und Filesharing

Wolfgang Finze
Ingenieurhochschule für Seefahrt Warnemünde/Wustrow

Filesharing bedeutet, daß mehrere Programme *gleichzeitig* auf eine Datei zugreifen. Solche gemeinsamen Zugriffe können bei der Anwendung von Rechnernetzen oder beim Einsatz von Multitask-Ergänzungen /1/ des Betriebssystems auftreten.
Typische Einsatzfälle einer Multitask-Ergänzung wären beispielsweise:
- Rechnernetze,
- Meßdatenerfassung mit paralleler Verarbeitung der eingegangenen Meßdaten,
- Einsatz von PCs für Steuerungsaufgaben im weitesten Sinne

Das Betriebssystem MS-DOS erlaubt ab Version 3.10 eine Dateiverarbeitung mit Filesharing. Allerdings muß, bevor das Betriebssystem solche gemeinsamen Zugriffe koordiniert, das DOS-Dienstprogramm SHARE /2/ geladen werden. Erst dieses speicherresidente Programm enthält die Routinen für die Koordination des Filesharing und auch für die Sperrung von Bereichen einer Datei gegen Mehrfachzugriff (Record Locking).
Ob eine Datei im Filesharing-Modus verarbeitet werden kann, wird bei ihrer Eröffnung festgelegt. Die Datei muß dazu mit der XENIX-kompatiblen DOS-Funktion 61 (3DH, Open Handle) eröffnet werden. Diese Funktion setzt voraus, daß die zu eröffnende Datei existiert. Existiert die Datei nicht, endet die Funktion 61 mit einer Fehlermeldung. Damit kann prinzipiell nur eine existierende Datei im Filesharing-Modus bearbeitet werden.
Die CP/M-kompatiblen DOS-Funktionen zur Dateiverwaltung erlauben keinen Mehrfachzugriff auf Dateien /3/ /4/ /5/.
Die Art der möglichen Zugriffe auf eine Datei wird durch den beim Aufruf der DOS-Funktion 61 im LOW-Teil des AX-Registers (AL) übergebenen Access-Code bestimmt. Dieser 1 Byte lange Wert setzt sich aus drei Bestandteilen zusammen, dem Inheritance Bit, dem Sharing- und dem Access-Mode, die jeweils durch bestimmte Bits im Access-Codes gebildet werden. Die Zuordnung der Bits des Access-Codes zu den Bestandteilen zeigt die folgende Abbildung /3/, /4/, /5/:



Die einzelnen Teile des Access-Codes haben folgende Bedeutung /4/:
Das **Inheritance-Bit** (Vererbungsbit), gibt an, ob die geöffnete Datei auch für einen Kindprozeß offen sein soll (Inheritance, Bit = 0) oder nicht (private, Bit = 1). Kindprozesse, die Dateien *erben* (inherit), übernehmen auch alle für diese Dateien geltenden Einschränkungen für File-Sharing und Dateizugriff.
Der **Sharing-Modus** gibt an, wie auf die geöffnete Datei parallel zugegriffen werden darf. Folgende Modi sind möglich:
Compatible: Kompatibilitätsmodus für Single-User-Systeme. Er erlaubt keinem anderen User ein Öffnen der Datei. Bitbelegung 000
Deny Both: Lesen und Schreiben für andere Nutzer verboten. Bitbelegung 001
Deny Write: Ein anderer Nutzer darf die Datei nicht zum Schreiben eröffnen. Bitbelegung 010
Deny Read: Ein anderer Nutzer darf die Datei nicht zum Lesen eröffnen. Bitbelegung 011
Deny None: Für andere Nutzer bestehen keine Einschränkungen bei der Verwendung der Datei. Bitbelegung 100
Der Access-Mode (Zugriffsmodus) gibt an, wie der eröffnende Prozeß selbst die Datei bearbeiten will. Folgende Zugriffsmodi sind möglich:
Read Only: Der aktuelle Prozeß will nur lesend auf die Datei zugreifen. Bitbelegung 0000
Write Only: Der aktuelle Prozeß will nur schreibend auf die Datei zugreifen. Bitbelegung 0001
Read/Write: Der aktuelle Prozeß will lesend und schreibend (Update-Modus) auf die Datei zugreifen. Bitbelegung 0010
Wenn eine Datei von einem Prozeß geöffnet wurde, wird die Möglichkeit, daß ein zweiter Prozeß parallel auf diese Datei zugreift, sowohl vom Access-Code des ersten als auch vom Access-Code des zweiten Prozesses bestimmt. Folgende Fälle können dabei auftreten:
- Die Datei wurde vom ersten Prozeß mit dem Sharing-Mode Compatible eröffnet. Zu beachten ist, daß dieser Sharing-Mode automatisch gesetzt wird, wenn eine Datei mit der DOS-Funktion 60 (Create Handle) oder den CP/M-ähnlichen DOS-Funktionen eröffnet wird.

Eine solche Datei kann von einem einzelnen Prozeß beliebig oft geöffnet werden, es sei denn, die Datei wurde bereits unter einem anderen der 4 Sharing-Modi eröffnet. Nur wenn die Datei das Attribut Read Only hat und mit dem Sharing-Mode Deny Write im lesenden Zugriff bearbeitet wird, kann mit lesendem Zugriff im Sharing-Mode Compatible zugegriffen werden.
- Die Datei wurde mit einem anderen als dem Modus Compatible geöffnet. Dabei können die in Tafel 1 dargestellten Fälle auftreten.

In Turbo-Pascal ist eine eigene, pascal-spezifische Dateiverwaltung implementiert, die sich auf die Betriebssystemfunktionen 3CH bis 42H stützt, wobei die Zuordnung der Prozeduren zu den Betriebssystemfunktionen wie folgt ist:

```
REWRITE  3CH – Create Handle
RESET    3DH – Open Handle
CLOSE    3EH – Close Handle
READ     3FH – Read Handle
WRITE    40H – Write Handle
SEEK     42H – Move File Pointer
```

Zur Dateiverwaltung ist in der Unit DOS der Typ FileRec vordefiniert. Er hat für typgebundene und typfreie Dateien folgenden Aufbau /6/:

```
TYPE FileRec = RECORD
        Handle:   Word;  { DOS-Handle }
        Mode:     Word;  { Modus      }
        RecSize:  Word;  { Recordgröße }
        Private:  array/1..26/ of Byte;
        UserData: array/1..16/ of Byte
        Name:     array/0..79/ of Char;
     END;
```

Für jede von Turbo-Pascal verwaltete typgebundene oder typfreie Datei existiert ein Record mit diesem Aufbau. Er wird durch die Prozedur ASSIGN angelegt und mit folgenden Werten gefüllt.

```
Handle  : 0
Mode    : 55216
   ( * gleich D7B0H = fmClosed * )
RecSize: 0
Name    : Dateiname
```

Der Dateiname ist dabei nicht in dem für Turbo-Pascal üblichen Stringformat abgelegt, sondern als sogenannter ASCIIZ-String, als eine Zeichenkette, die kein Längenbyte besitzt, sondern vollständig aus ASCII-Zeichen besteht und deren Ende durch das Zeichen ∧@ (Hexadezimal 00) bestimmt wird. ASCIIZ-Strings sind nicht kompatibel mit den Strings aus Turbo-Pascal.
Die Größe *FileRec.Mode* kann nur einen der folgenden 4 Werte annehmen /4/:

```
fmClosed = D7B0H;
fmInput  = D7B1H;
fmOutput = D7B2H;
fmInOut  = D7B3H;
```

Diese Werte sind in der Unit DOS als Konstanten vordefiniert, wobei fmInput und fmOutput nur für Textdateien Bedeutung haben. Für typgebundene oder typfreie Dateien

Tafel 1 Verträglichkeit von Share-Modi

Prozeß 1 eröffnete mit	Prozeß 2 kann öffnen mit
Deny Both, Read Only (AL = x0010000)	kann nicht geöffnet werden!
Deny Both, Write Only (AL = x0010001	kann nicht geöffnet werden!
Deny Both, Read/Write (AL = x0010010)	kann nicht geöffnet werden!
Deny Write, Read Only (AL = x0100000)	Deny Write, Read Only (AL = x0100000) Deny None, Read Only (AL = x1000000)
Deny Write, Write Only (AL = x0100001)	Deny Read, Read Only (AL = x0110000) Deny None, Read Only (AL = x1000000)
Deny Write, Read/Write (AL = x0100010)	Deny None, Read Only (AL = x1000000)
Deny Read, Read Only (AL = x0110000)	Deny Write, Write Only (AL = x0100001) Deny None, Write Only (AL = x1000001)
Deny Read, Write Only (AL = x0110001)	Deny Read, Write Only (AL = x0110001) Deny None, Write Only (AL = x1000001)
Deny Read, Read/Write (AL = x0110010)	Deny None, Write Only (AL = x1000001)
Deny None, Write Only (AL = x1000001)	Deny Read, Read Only (AL = x0110000) Deny Read, Write Only (AL = x0110001) Deny Read, Read/Write (AL = x0110010) Deny None, Read Only (AL = x1000000) Deny None, Write Only (AL = x1000001) Deny None, Read/Write (AL = x1000010)
Deny None, Read/Write (AL = x1000010)	Deny None, Read Only (AL = x1000000) Deny None, Write Only (AL = x1000001) Deny None, Read/Write (AL = x1000010)

wird beim Eröffnen immer der Wert fmInOut eingetragen. Bei einem anderen als diesem Wert wird von Turbo-Pascal jeder Zugriff auf die Datei verweigert /6/.

Für die Verarbeitung von Textdateien existiert ebenfalls ein dateibeschreibender Satz, der aber anders aufgebaut ist. Der genaue Aufbau kann in /6/ nachgelesen werden, hier soll auf Textdateien nicht weiter eingegangen werden.

Die Prozeduren zum Anlegen, Öffnen oder Schließen einer Datei (REWRITE, RESET, CLOSE) greifen auf den dateibeschreibenden Satz zu und aktualisieren den Inhalt der Felder Handle, Mode und RecSize.

Bei Ausführung von RESET oder REWRITE werden Dateinummer (Handle) und Satzlänge eingetragen und der Wert *FileRec-.Mode* wird auf D7B3H gesetzt. Ein CLOSE ändert im dateibeschreibenden Satz nur den Mode auf D7B0H und läßt alle anderen Werte unverändert.

Bei der Eröffnung einer typgebundenen oder typfreien Datei mit **RESET** (DOS-Funktion 3DH) wird der LOW-Teil des AX-Registers (AL) durch den Inhalt der in der Unit **SYSTEM** definierten typgebundenen Konstanten *FileMode* bestimmt. Dieser Wert bestimmt den für die Datei gültigen Access-Code und hat den voreingestellten Wert 2, so daß eine Datei bei den Versionen 4 und 5 von Turbo-Pascal im Sharing-Mode *Compatible* und mit dem Zugriffscode **READ/WRITE** eröffnet wird.

Es ist möglich, diese Voreinstellung über eine Wertzuweisung an die Größe FileMode zu ändern, so daß in Turbo-Pascal eine typgebundene oder typfreie Datei mit einem beliebigen Access-Code eröffnet werden kann.

Die Größe FileMode bezieht sich nicht auf eine bestimmte Datei, sondern der FileMode zugewiesene Wert wird bei allen folgenden Dateieröffnungen mit **RESET** verwendet. Wenn mehrere Dateien mit unterschiedlichem Access-Code eröffnet werden sollen, ist also jedesmal **vor** der Eröffnung der Datei der Größe FileMode ein entsprechender Wert zuzuweisen.

Die beiden folgenden kurzen Beispielprogramme bearbeiten gemeinsam die gleiche Datei. Während das Programm multi_1 eine bestehende Datei korrigiert und erweitert, liest das Programm multi_2 parallel dazu diese Datei (Bild 1 und 2).

Beide Beispiele liefen parallel unter einer multitaskfähigen Erweiterung des Betriebssystems MS-DOS, wobei das Programm multi_2 nach dem Programm multi_1 gestartet wurde. Die Verzögerungsschleifen in den Beispielen dienen nur dazu, die Arbeit der Programme optisch besser verfolgen zu können.

Es ist also möglich, mit den in Turbo-Pascal vordefinierten Sprachelementen zur Dateiverarbeitung (Read, Write, BlockRead, BlockWrite, Seek, FileSize, FilePos, EOF, IOResult ...) Dateien mit File-Sharing zu bearbeiten, das heißt Programme zu schreiben, in denen ein gleichzeitiger Mehrfachzugriff auf Dateien erlaubt ist.

Sprachelemente zur satzweisen Verriegelung von Dateien (Lock und Unlock) sind in den Versionen 4 und 5 von Turbo-Pascal allerdings **nicht** vordefiniert.

Wenn beispielsweise für den Aufbau eines Netzwerkes solche Sprachelemente benötigt werden, müssen sie unter Verwendung der entsprechenden DOS-Funktionen implementiert werden. Eine Möglichkeit der satzweisen Verriegelung und Entriegelung soll hier vorgestellt werden.

Zusätzlich wird noch die Funktion IOCTL-_Retry = DOS-Funktion 44H zur Einstellung der Anzahl der auszuführenden Zugriffe bei einem Sharing-Konflikt eingefügt.

MS-DOS stellt zur satzweisen Ver- und Entriegelung die Funktion 92 (5CH) bereit. Ein im AL-Register übergebener Code regelt, ob Teile einer Datei gegen jeglichen Zugriff eines anderen Prozesses gesperrt werden sollen (**Lock**) oder ob eine gesetzte Zugriffssperre wieder aufgehoben werden soll (**Unlock**). Die Funktion 92 arbeitet nur in Verbindung mit dem Dienstprogramm **SHARE**.

Folgende Register müssen bei Aufruf der Funktion belegt sein:

AH :5CH
AL :00H – Verriegeln eines Bereiches (Lock)
01H – Entriegeln eines Bereiches (Unlock)
BX: Handle der Datei
CX:DX : Position (Offset, in Byte) des ersten zu sperrenden oder freizugebenden Bytes. Gezählt wird von Dateibeginn an.
SI:DI: Länge (in Byte) des zu sperrenden oder freizugebenden Bereiches.

Bei Nutzung dieser Funktion ist folgendes zu beachten /1/ /3/:

- Falls eine geöffnete Datei an einen Kindprozeß übergeben wird, gelten eventuelle Sperrungen für den Kindprozeß **nicht**.
- Es ist möglich, auch Dateibereiche nach dem Dateiende zu sperren. Das ist insbesondere für das Erweitern einer Datei von Bedeutung.
- Eine Sperrung sollte immer nur *vorübergehend* sein und immer explizit aufgehoben werden. Insbesondere sollten beim Schließen der Datei **keine** Bereiche mehr gesperrt und im Programm dafür gesorgt sein, daß bei einem Programmende mit den Interrupts 23H oder 24H eventuelle Sperrungen aufgehoben werden. Das bedeutet, die Programme sollten eigene Handler für die genannten Interrupts haben.
- Es können nur Bereiche entriegelt werden, die vom gleichen Prozeß auch verriegelt wurden und die in Größe und Lage übereinstimmen, sonst wird der Fehler 21H (Lock Violation) gemeldet.
- Falls ein Bereich bereits gesperrt ist, wird beim Versuch einer erneuten Sperrung des Bereiches der Fehler 21H (Lock Violation) gemeldet, unabhängig davon, ob der eigene oder ein anderer Prozeß die Sperrung verursacht hat.

Greift ein anderer Prozeß auf einen gesperrten Dateibereich zu, tritt ein Fehler auf, da der Zugriff nicht ausgeführt wurde. Das Betriebssystem wiederholt einen solchen Zugriff noch dreimal. Erst wenn auch diese Wiederholungen zum Fehler führen, wird der Interrupt 24H ausgelöst. Die Anzahl der vom Betriebssystem auszuführenden Wiederholungen und die Zeit zwischen zwei Wiederholungen kann mit der Betriebssystemfunktion 44H, Unterfunktion 11 (IOCTL Retry) verändert werden.

```
{$M 16384,0,65000 }
program multi_1;
(*-------------------------------------------------------------*)
(* Demonstrationsprogramm 1 für  den Zugriff mehrerer Nutzer   *)
(* auf eine Datei                                              *)
(*-------------------------------------------------------------*)
(* Diese Programm schreibt in eine Datei, die vom Programm     *)
(* Multi_2 gelesen wird, waerend dieses Programm schreibt.     *)
(*-------------------------------------------------------------*)
CONST Vorsatz      = 'korrigierter Satz :';
TYPE  Inhalt       = String[35];
VAR   Satz         : Inhalt;
      I,j          : Word;
      Datei        : File of Inhalt;
      Nummer       : String[8];
BEGIN
  FileMode:=$20 + $02;  { Inherit, Deny write, Read/Write }
  ASSIGN(Datei,'TEST.DAT');
{$I-}
  RESET(Datei);
{$I+}
  IF IOResult <> 0 THEN
  WRITELN('Fehlercode beim Eroeffnen : ',IOResult)
  ELSE BEGIN
    SEEK(Datei,10);
    FOR i:=1 TO 50 DO BEGIN
      Str(i,Nummer);
      Satz:=Vorsatz+Nummer;
      WRITE(Datei,Satz);
      WRITELN('korrigiert : ',Satz);
      FOR j:=1 TO 64000 DO;  { Verzögerungsschleife  }
    END;
    CLOSE(Datei);
  END;
END.
```

◀ *Bild 1*

```
{$M 16384,0,65000 }
program multi_2;
{--------------------------------------------------------------}
{ Demonstrationsprogramm 2 für  den Zugriff mehrerer Nutzer auf }
{ eine Datei                                                   }
{ Die von MULTI_1 korrigierte Datei wird "mitgelesen"          }
{--------------------------------------------------------------}
{   Sharing-Mode  Deny none   -  Zugriffsmodus     read        }
{--------------------------------------------------------------}
TYPE  Inhalt           = String[35];
VAR   Satz             : Inhalt;
      Datei            : File of Inhalt;
      I,j, gelesen     : Word;
      Mode             : Byte;

BEGIN
  FileMode:=$40;          { Inherit, Deny none, Read  }
  ASSIGN(Datei,'TEST.DAT');
{$I-}
  RESET(Datei);
{$I+}
  IF IOResult <> 0 THEN
  WRITELN('Fehlercode beim Eroeffnen : ',IOResult)
  ELSE BEGIN
    REPEAT
      READ(Datei,Satz);
      IF NOT EOF(Datei) THEN WRITELN('gelesen : ',Satz);
      FOR j:=1 TO 64000 DO;   { Verzögerungsschleife  }
    UNTIL EOF(Datei);
    CLOSE(Datei);
  END;
END.
```

Bild 2

Diese Funktion benötigt beim Aufruf folgende Registerbelegung /1/ /3/ /4/:

AH: 44H
AL: 0BH
BX: gewünschte Anzahl von Wiederholungen des Zugriffs
CX: Zeit, die zwischen zwei Versuchen gewartet werden soll.

Bei der Nutzung dieser Funktion ist zu beachten, daß die Angabe der Zeit, die zwischen

Bild3

```
UNIT LOCKING;
{--------------------------------------------------------------------}
{  Unterstützung der Dateiarbeit in Multiuser- und Multitask-        }
{  Systemen.                                                         }
{--------------------------------------------------------------------}
{                  Parameterbeschreibung                             }
{                                                                    }
{   M_Dat          : Name der zu bearbeitenden Datei                 }
{   pos            : Nummer des zu verriegelnden Satzes              }
{   Wiederholungen : Anzahl der Zugriffsversuche, bevor im Kon-      }
{                    fliktfall eine Fehlermeldung (Interr. $24)      }
{                    ausgegeben wird, Standard : 3                   }
{   Wartezeit      : Wartezeit zwischen zwei Zugriffsversuchen       }
{   ErCode         : Fehlercode                                      }
{--------------------------------------------------------------------}
{ W.Finze, IH für Seefahrt Warnemünde, Direktorat WGB                }
{           Wissenschaftsbereich Informationsverarbeitung            }
{ Version : 2.0              01.12.1988                              }
{--------------------------------------------------------------------}

INTERFACE
USES DOS;
CONST _Inherit = $00;          { Inheritance-Bit Inherit          }
      _Private = $80;          { Inheritance-Bit Private          }
      _Comp    = $00;          { Sharing-Mode Compatibility       }
      _DRW     = $10;          { Sharing-Mode Deny both           }
      _DW      = $20;          { Sharing-Mode Deny write          }
      _DR      = $30;          { Sharing-Mode Deny read           }
      _DN      = $40;          { Sharing-Mode Deny none           }
      _R       = $00;          { Access-Code Read                 }
      _W       = $01;          { Access-Code Write                }
      _RW      = $02;          { Access-Code Both                 }

procedure M_Retry(Wiederholungen, Wartezeit : Word;
                  Var ErCode                : Word);
procedure M_Lock(Var M_Dat; pos : Word;
                 Var ErCode     : Word);
procedure M_Unlock(Var M_Dat; pos : Word;
                   Var ErCode     : Word);

IMPLEMENTATION
{--------------------------------------------------------------------}
Var Regs   : Registers;
    Struct : ^FileRec;

procedure M_Retry(Wiederholungen, Wartezeit : Word;
                  Var ErCode                : Word);
begin
  Regs.AH:=$44;
  Regs.AL:=$0B;
  Regs.BX:=Wiederholungen;
  Regs.CX:=Wartezeit;
  MSDOS(Regs);
  IF (Regs.Flags AND $0001 ) = 1 then ErCode:=Regs.AX;
end;

procedure M_Lock(Var M_Dat; pos : Word;
                 Var ErCode     : Word);
Var NewPos : LongInt;
    Words  : Array[1..2] of Word ABSOLUTE NewPos;
begin
  Struct:=@M_Dat;
  with Struct^ do begin
    NewPos:=RecSize;
    NewPos:=NewPos*Pos;
    Regs.AX:=$5C00;
    Regs.BX:=Handle;
    Regs.CX:=Words[2];    { HI-Wort Anfang der Verriegelung }
    Regs.DX:=Words[1];    { LO-Wort Anfang der Verriegelung }
    Regs.SI:=0;           { HI-Wort Recordlaenge            }
    Regs.DI:=RecSize;     { LO-Wort Recordlaenge            }
    MSDOS(Regs);
    IF (Regs.Flags AND $0001) = 1 then ErCode:=Regs.AX;
  end;
end;

procedure M_Unlock(Var M_Dat; pos : Word;
                   Var ErCode     : Word);
Var NewPos : LongInt;
    Words  : Array[1..2] of Word ABSOLUTE NewPos;
begin
  Struct:=@M_Dat;
  with Struct^ do begin
    NewPos:=RecSize;
    NewPos:=NewPos*Pos;
    Regs.AX:=$5C01;
    Regs.BX:=Handle;
    Regs.CX:=Words[2];    { HI-Wort Anfang der Entriegelung }
    Regs.DX:=Words[1];    { LO-Wort Anfang der Entriegelung }
    Regs.SI:=0;           { HI-Wort Recordlaenge            }
    Regs.DI:=RecSize;     { LO-Wort Recordlaenge            }
    MSDOS(Regs);
    IF (Regs.Flags AND $0001) = 1 then ErCode:=Regs.AX;
  end;
end;
{--------------------------------------------------------------------}
END.
```

Bild 4

```
{$M 16384,0,65000 }
program Multi_4;
(*---------------------------------------------------------------*)
(* Demonstrationsprogramm 4 fuer den Zugriff mehrerer Nutzer   *)
(* auf eine Datei                                               *)
(*---------------------------------------------------------------*)
(* Diese Programm schreibt eine Datei, die vom Programm         *)
(* Multi_5 korrigiert wird, waerend dieses Programm schreibt. *)
(* Die Anzahl der Wiederholungen wird auf 30 gesetzt, die Zeit*)
(* auf 100                                                      *)
(*---------------------------------------------------------------*)

uses Locking;
Const Vorsatz      ='Ursprungssatz :';
      Verzoegerung = 64000;
TYPE  Inhalt       = String[35];
VAR   Satz         : Inhalt;
      I,j,NewPos,
      geschrieben  : Word;
      Datei        : File of Inhalt;
      Nummer       : String[8];
      Fehler       : Word;
begin
  Assign(Datei,TEST.DAT');
  M_Retry(30,100,Fehler);
  Rewrite(Datei);
  Close(Datei);
  FileMode:=_Inherit + _DN + _RW;  { Inherit, Deny none, Read/Write }
{$I-}
  Reset(Datei);
{$I+}
  if IOResult <> 0 then
  writeln('Fehlercode beim Eroeffnen : ',IOResult)
  else begin
    NewPos:=1;
    for i:=1 to 50 do begin
      Str(i,Nummer);
      Satz:=Vorsatz+Nummer;
      M_Lock(Datei,NewPos,Fehler);  { Verriegeln nach EOF }
      Write(Datei,Satz);
      M_Unlock(Datei,NewPos,Fehler); { Entriegeln        }
      NewPos:=NewPos+1;
      writeln('geschrieben : ',Satz);
      for j:=1 to Verzoegerung do;
    end;
    Close(Datei);
  end;
end.
```

Bild 5

```
{$M 16384,0,65000 }
program multi_5;
(*---------------------------------------------------------------*)
(* Demonstrationsprogramm 5 fuer den Zugriff mehrerer Nutzer   *)
(* auf eine Datei                                               *)
(*---------------------------------------------------------------*)
(* Diese Programm liest und korrigiert eine Datei, waerend     *)
(* diese Datei vom Programm Multi_4 erstellt wird.             *)
(*---------------------------------------------------------------*)
uses Locking;
Const Vorsatz      ='korrigierter Satz :';
      Verzoegerung = 32000;
TYPE  Inhalt       = String[35];
VAR   Satz         : Inhalt;
      I,j,
      gelesen,
      geschrieben  : Word;
      Datei        : File of Inhalt;
      Name         : String[80];
      Nummer       : String[8];
      Fehler       : Word;
      NewPos       : Word;

begin
 FileMode:=_Inherit+_DN+_RW; { Inherit, Deny none, Read/Write }
  M_Retry(30,250,Fehler);
  assign(Datei,'TEST.DAT');
{$I-}
  Reset(Datei);
{$I+}
  i:=1;
  if IOResult <> 0 then
  writeln('Fehlercode beim Eroeffnen : ',IOResult)
  else begin
    repeat
      Read(Datei,Satz);     { Satz lesen  }
      If IOResult <>0 then writeln('Fehler beim Lesen, Wert      : ',
                                   IOResult);
      If EOF(Datei) then Writeln('Dateiende nach  ',(i-1):2,
                                 '-tem Satz erreicht !');
      if not EOF(Datei) then begin
        writeln('alter Inhalt : ',Satz);
        Str(i,Nummer);
        Satz:=Vorsatz+Nummer;
        Seek(Datei,i-1);       { Dateizeiger um 1 zurueck }
    { ---- verriegeln, schreiben, entriegeln --- }
        M_Lock(Datei,i-1,Fehler);
        Write(Datei,Satz);
        If IOResult <>0 then writeln('Fehler beim Schreiben, Wert : ',
                                     IOResult);
        M_Unlock(Datei,i-1,Fehler);
        writeln('neuer Inhalt : ',Satz);
        for j:=1 to Verzoegerung do;
        i:=i+1;
      end;
    until EOF(Datei);
    Close(Datei);
  end;
end.
```

zwei Versuchen gewartet werden soll, keine absolute Größe ist. Festgelegt wird eigentlich nur, wie oft eine Warteschleife zu durchlaufen ist /4/. Die für diese Schleife benötigte Zeit ist vom Prozessor und von der Taktfrequenz abhängig, so daß hier keine Optimalwerte mitgeteilt werden können. Solche Werte müssen immer für das jeweilige System experimentell ermittelt werden.
Zur Realisierung der gewünschten Funktionen wurde die Unit **LOCKING** aufgebaut (Bild 3). Zur Vereinfachung der Arbeit mit dem Access-Code und den Dateiattributen wurden neben den genannten Prozeduren in der Unit Konstanten vordefiniert, die es erlauben, die jeweiligen Werte für FileMode durch Addition von Konstanten zu bilden.
Bei der Realisierung wurde auf den dateibeschreibenden Satz, der von Turbo-Pascal angelegt wird, zurückgegriffen. Diesem Satz wurde das Handle der Datei und die aktuelle Satzlänge entnommen. Die Prozeduren zum Ver- oder Entriegeln wurden in Anlehnung an die in Turbo-Pascal übliche Dateiarbeit so aufgebaut, daß Lock und Unlock nur für einen ganzen Datensatz möglich sind, obwohl die DOS-Funktionen prinzipiell sogar das Ver- oder Entriegeln einzelner Bytes erlauben.
Wenn mit File-Sharing gearbeitet wird, ist die folgende Besonderheit auch dann zu beachten, wenn keine Dateibereiche verriegelt sind. Falls eine Datei nicht ordnungsgemäß geschlossen wurde (z. B. nach einem Programmabsturz), kann es sein, daß sich diese Datei nicht mit Betriebssystemkommandos löschen läßt. Jeder Versuch einer Löschung wird mit der Fehlermeldung **Allgemeiner Diskettenfehler** abgewiesen. Vor einer Löschung ist entweder das Programm **SHARE** aus dem Speicher zu entfernen, oder die Datei muß von einem Programm erneut geöffnet **und** geschlossen werden.
Die beiden folgenden Beispielprogramme verwenden die oben beschriebene Unit Lokking (Bild 4 und 5).
Da das Programm multi_5 einen geringeren Verzögerungsfaktor besitzt, arbeitet es schneller als das Programm multi_4. Multi_5 erreicht daher das Dateiende, während multi_4 noch schreibt. In diesem Fall wird im Programm multi_5 die EOF-Bedingung *wahr*, und das Programm multi_5 beendet seine Arbeit, während multi_4 die Dateikorrektur fortsetzt.
Es sei abschließend noch einmal darauf hingewiesen, daß die parallele Abarbeitung mehrerer Programme **nur** mit einer multitaskfähigen Betriebssystemerweiterung (z. B. SM-DOS, Deskview oder Multilink) möglich ist, da MS-DOS selbst nicht multitaskfähig ist.

Literatur

/1/ Kalfa, W.: Software für 16-Bit-Systeme – Teil 1 Das Betriebssystem DCP. Schriftenreihe Informationsverarbeitung im Hoch- und Fachschulwesen, Heft 15 Berlin: Zentralinstitut für Hoch- und Fachschulwesen 1988

/2/ DOS 3.30 – Betriebssystem DOS, Referenzhandbuch. IBM Deutschland GmbH, 1987

/3/ MS-DOS 3.1 Programmierhandbuch in englischer Sprache/Microsoft. – Haar bei München: Markt & Technik-Verlag 1986

/4/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. München: Francis-Verlag, 1987

/5/ Norton, P.: Programmierhandbuch für den IBM-PC. Braunschweig: Vieweg & Sohn Verlagsgesellschaft, 1986

/6/ Turbo-Pascal 4.0 Referenz- und Benutzerhandbuch. München: Heimsoeth & Borland, 1987

✉ KONTAKT ☏

Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Direktorat Wissenschaftlicher Gerätebau, Wissenschaftsbereich Informationsverarbeitung, Richard-Wagner-Str. 31, Warnemünde, 2530

Floppy-Treiber mit FDC U 8272 im Nicht-DMA-Modus   Teil 1

Dr. Christian Löber
VEB Robotron Meßelektronik „Otto Schön" Dresden

Der Einsatz des Floppy-Disk-Controllers U 8272 ermöglicht eine wesentliche Verringerung des Hardware- und des Softwareaufwandes bei der Ansteuerung von Folienspeichern. Wird auf den Anschluß von 8"-Laufwerken (doppelte Schreibgeschwindigkeit) verzichtet, so kann der FDC im Nicht-DMA-Modus sowie im reinen Pollingverfahren (ND-Poll-Prinzip) betrieben und somit der Aufwand für FD-Controller und FD-Treiber weiter reduziert werden. Der folgende Bericht beschreibt den CP/M-kompatiblen Treiber für eine Diskettenstation mit ein bis drei Laufwerken vom Typ K 5601 (MFS 1.6). Es wird die Kenntnis der Grundfunktionen des FDC U 8272 vorausgesetzt /1/, /2/.

4-MHz-Version FDC U 8272 D04

Der FDC U 8272 unterstützt die doppelseitige Aufzeichnung von Datenströmen auf softsektorierte Disketten nach dem standardisierten Datenträgerformat gemäß KROS-Standard 5110/01 /3/.
Der Spuraufbau ist gekennzeichnet durch eine programmierbare, am Indexloch beginnende Anzahl von Sektoren pro Spur. Jeder Sektor besteht aus dem Kennzeichnungsfeld (ID-Feld) und dem eigentlichen Datenfeld, eingerahmt durch eine feste bzw. formatabhängige Anzahl von Lückenbytes.
Bei einer Umdrehungsgeschwindigkeit von 5/s und einem Schreibtakt von 4 μs/Bit ($\triangleq$ 32 μs/Byte) enthält die Spur einer Minidiskette maximal 6250 Byte mit optimal $5 \times 1024 = 5120$ Nutzbytes. Bei doppelseitiger Aufzeichnung auf 80 Spuren ergibt dies eine maximale Speicherkapazität von 800 KByte. Jedes andere Format (z. B. 9×512, 16×256) reduziert die zur Verfügung stehende Kapazität.

Grundstruktur

Der U 8272 enthält unter anderem eine parallele Schnittstelle zum Prozessorbus, eine serielle Schnittstelle zum Anschluß an das Laufwerksinterface sowie ein Hauptstatusregister und ein Datenregister zur Steuerung des Datenaustausches. Zur Abwicklung der Datenübertragung stehen 15 FDC-Befehle zur Verfügung. Die Befehle sind im allgemeinen durch drei Phasen charakterisiert:
Befehlsphase
In dieser Phase versorgt der Prozessor den FDC mit den aktuellen Parametern einer durchzuführenden Operation.
Ausführungsphase
Die angewiesene Operation wird vom FDC ausgeführt. Bei Sektoroperationen werden die Daten im DMA-Modus am Prozessor vorbei über den DMA-Schaltkreis direkt zwischen dem aktuellen RD/UR-Puffer und dem FDC-Datenregister übertragen. Im Nicht-DMA-Modus erfolgt das Bytehandling voll programmgesteuert über den Prozessor mit dem FDC. Dieser Ablauf unterliegt harten Zeitbedingungen, da als Transferzeit für die Datenströme zwischen dem FDC und der Diskette nur maximal 32 μs pro Byte zur Verfügung stehen.
Ergebnisphase
In dieser Phase stellt der FDC je nach Befehl bis zu 7 Resultatbytes – darunter die Statusinformationen aus den Stapelregistern ST0 ... ST3 – über das Ergebnis der Operation zur Verfügung.
Jedes Resultat muß vollständig Byte für Byte gelesen werden, bevor eine neue Operation befohlen werden kann.

Hauptstatusregister FDCSTAT

Das nur lesbare Hauptstatusregister stellt Basisinformationen insbesondere für die zeitliche Synchronisation Prozessor–FDC bereit (Bild 2). Dabei sind die Bits B4 ... B7 speziell bei einem im Nicht-DMA-Verfahren und im reinen Pollingverfahren arbeitenden Treiber von besonderer Bedeutung, da hierbei der gesamte zeitliche Ablauf, von wenigen Ausnahmen abgesehen, ausschließlich über die Statusinformation dieser vier Bits abgewickelt wird.
DB0 ... DB3
Diese Bits kennzeichnen, daß das betreffende Laufwerk im Suchbetrieb ist. Es ist zu beachten, daß sie erst durch Ausgabe des Befehls SENSE INTERRUPT STATUS zurückgesetzt werden.

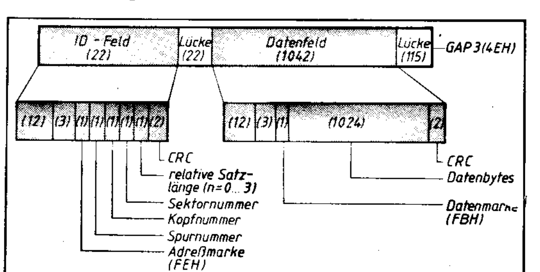
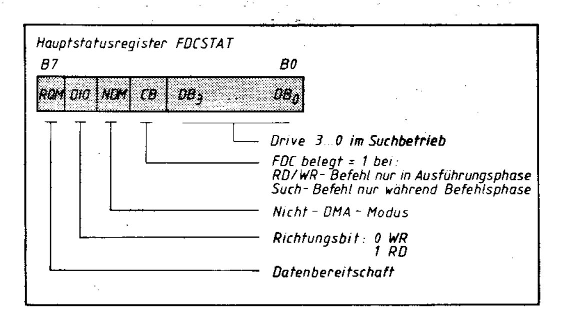
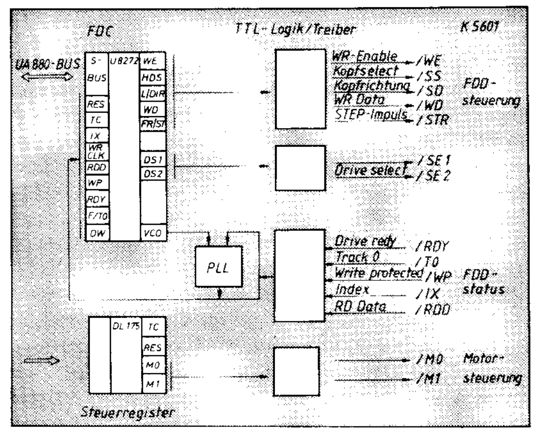


Bild 1 Struktur eines 1024-Byte-Sektors einer Minidiskette (in Klammern Anzahl der Bytes)

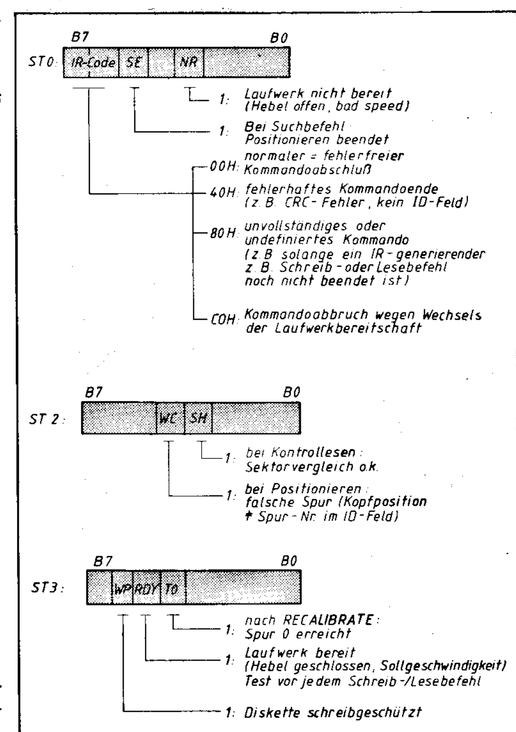


◀ **Bild 2 Bitbelegung des Hauptstatusregisters**



◀ **Bild 4 Blockschaltbild FD-Controller mit FDC U 8272 D04 im ND-Poll-Verfahren**

Bild 3 Statusinformationen für das Pollingverfahren ▶



CB
Ist gesetzt während der Ausführungsphase eines Lese- oder Schreibkommandos, bei Suchbetrieb im Unterschied zu DB0...DB3 jedoch nur während der Befehlsphase des RECALIBRATE- bzw. SEEK-Befehls, um so das zeitlich parallele Positionieren mehrerer Laufwerke zu ermöglichen. B4 kann beispielsweise bei der Abholung des Resultats in der Ergebnisphase eines Befehls dazu benutzt werden, zu erkennen, ob vom FDC weitere Resultatbytes abzuholen sind (CB ist dann noch = 1).

ND
Charakterisiert den Nicht-DMA-Modus und ist dabei während der Ausführungsphase einer Operation High. Dadurch kann das gegebenenfalls vorzeitige oder unbeabsichtigte Ende einer Operation im ND-Modus erkannt werden. Das ist beispielsweise der Fall, wenn bei Sektorlese- oder Schreiboperationen vom FDC kein gültiges ID-Feld gefunden wird (defekte Spur, unformatierte Diskette, unzeitgemäßes Öffnen des Laufwerkhebels). Der FDC meldet dann bei SEC-RD-Operationen anstelle des Sollwertes F0H den Wert D0H, das heißt, B7, B6 und B4 haben zwar Sollstatus, B4 = 0 jedoch nicht. Bei unsachgemäßer Programmierung des FD-Treibers (z. B. keine Zeitüberwachung in der Bytehandling-Schleife) bleibt dieser dann in einer Endlosschleife hängen (vgl. auch Programmbeispiel in Bild 11). Gleiches gilt für das Bytehandling bei SEC-WR-Operationen. Dabei wird anstelle des Sollmusters B0H (hier DIO-Bit = 0 wegen geänderter Übertragungsrichtung) der Wert 90H gemeldet.

DIO
Charakterisiert die Übertragungsrichtung:
0: CPU = => FDC (Schreiben)
1: CPU <= = FDC (Lesen)
Nach RESET ist DIO = 0.

RQM
Kennzeichnet die Datenbereitschaft des FDC. Nur wenn RQM = 1, ist der FDC bereit, ein Byte vom Prozessor ins Datenregister zu übernehmen oder aus diesem an den Prozessor abzugeben. Auf die Datenbereitschaft des FDC muß bei der 4-MHz-Version U 8272 D04 innerhalb von 24 μs reagiert werden. RQM ist beim ND-Poll-Verfahren das wichtigste Synchronisationsbit.

Datenregister FDCDAT

Das Datenregister ist lesbar und schreibbar. Es steht mit einem Stapel von 7 Pufferregistern in Verbindung, die über das Datenregister – speziell in der Resultatsphase eines Befehls – der Reihe nach ausgelesen werden können.
Über FDCDAT erfolgt:

– die Programmierung des FDC für ein bestimmtes Kommando einschließlich der Übergabe der zugehörigen Laufwerksparameter (z. B. Code für Spursuche, Spurnummer usw.)
– das Lesen bzw. Schreiben der Sektordaten (ID-Feld, Datenfeld)
– das Rücklesen des Ergebnisses einer Operation aus den Statusregistern ST0...ST3 (fehlerfreie bzw. fehlerhafte Kommandoausführung) sowie des Istzustandes der Laufwerksparameter.

Die Statusregister ST0...ST3 liefern für die Programmierung eines konkreten FD-Treibers eine überbestimmte Menge an Statusinformationen. Für einen im Polling-Verfahren arbeitenden Treiber genügt neben den Informationen aus dem Hauptstatusregister die in Bild 3 angegebene Untermenge zur Synchronisation des gesamten Ablaufs.
Dabei sind die Interruptbits B7 und B6 von ST0 von besonderer Bedeutung, da sie den Status über den erfolgreichen bzw. fehlerhaften Abschluß der befohlenen Operation enthalten.
Die Bits 3, 4 und 5 von ST3 liefern wichtige Vorinformationen über die Durchführbarkeit der Sektorarbeit, zum Beispiel ob das anzusprechende Laufwerk bereit ist (B5) oder ob die gesteckte Diskette schreibgeschützt ist (B6).
Die Register ST1 und ST2 liefern eine Reihe von Statusbits zur Charakterisierung aufgetretener Fehler, beispielsweise CRC-Fehler, schlechte Spur, Datenfehler, fehlende Adreßmarke usw. Praktisch ist ihre Auswertung bei CP/M-kompatiblen Betriebssystemen jedoch von untergeordneter Bedeutung, da im allgemeinen vom physischen Treiber außer *DRIVE NOT READY* oder *WRITE PROTECTED* an das logische E/A-System ohnehin nur der binäre Status: kein Fehler (A ≠ 0) oder Fehler (A = 0) zurückgegeben wird, woraus dann BDOS die für den *normalen* Bediener ausreichende Fehlermeldung *BAD SECTOR* generiert. Die Nutzung der detaillierten Fehlerangaben in ST1 und ST2 ist hingegen sehr nützlich beim Einfahren des Treibers und bei der Analyse von Funktionsstörungen.

Interruptverhalten

Der FDC erzeugt zwar bei Eintritt in die Resultatsphase verschiedener Kommandos und während der Ausführungsphase im Nicht-DMA-Betrieb sowie in einigen weiteren Fällen ein IR-Signal zur Unterstützung der Synchronisation der Ablaufsteuerung FDC–Laufwerk /2/, verfügt jedoch nicht über die für den vektorierten IR-Betrieb mit dem U 880-Prozessor erforderliche IR-Steuer- und Prioritätslogik. Allein schon deswegen ist ein reines Pollingverfahren im FD-Treiber zweckmäßig. Es gibt dafür aber noch weitere Gründe.

FD-Controller

Ein auf dem ND-Poll-Prinzip beruhender FD-Controller enthält neben der 4-MHz-Version U 8272 D04 des FDC weniger als ein Dutzend einfache, im wesentlichen nur als Treiber fungierende TTL-Schaltkreise (Bild 4). Ein zusätzliches Steuerregister DL 175 dient zum Generieren folgender Signale:
– Motorsteuerbits M0 und M1 für die Ansteuerung von maximal vier Laufwerken
– TC-(Terminal Count-)Signal (ein > 4 μs langer High-Impuls zum Beenden von Sektoroperationen)
– programmierbares FDC-RESET-Signal zum Rücksetzen der internen FDC-Register. Durch das SPEZIFY-Kommando program-

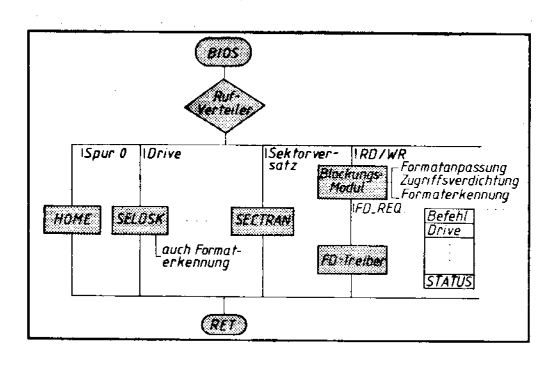


Bild 5 Software-Komponenten für die physische Diskettenarbeit bei CP/M-kompatiblen Betriebssystemen

Bild 6 Floppy-Requestvektor FD_REQ zwischen Blockungsmodul und FD-Treiber

FD_REQ: Befehlscode; Laufwerk; Spur; Kopf (Seite); Sektor; relative Datenfeldlänge; Lücke (GAP 3); Sektoranzahl; Verfahren (FM/MFM); Schritte/Spur; RD/WR-Puffer; Driveparameterblock; Status (Error)
n = 0...3 = 128, 256, 512, 1024 B
DRPB: Laufwerktyp; 1. Spur mit Präkompensation; Schrittzeit SRT; Kopfladezeit HLT; Kopfberuhigung nach STEP; 1. Spur m. Schreibstrombegrenz.; B3 = 0: mit Kontrollesen; Präkompensationszeit

mierte FDC-Parameter werden durch dieses Signal nicht gelöscht.
Des weiteren ist eine einfache digitale PLL-Schaltung zur Synchronisation der RD-Impulse mit dem Datenfenstersignal DW vorgesehen. Vom Laufwerksinterface werden die Statussignale /T0 = Spur 0 erreicht, /RDY = Laufwerk bereit, /WP = Diskette schreibgeschützt sowie /IX = Spuranfang an den FDC gemeldet, wobei die ersten drei über das Statusregister ST3 auch programmtechnisch zugänglich sind (Bild 3).
Auf Grund des ND-Poll-Verfahrens enthält die Schaltung des FD-Controllers weder IR- noch DMA-Steuerlogik.

Diskettenarbeit bei CP/M-kompatiblen Betriebssystemen

Die für die Diskettenarbeit erforderliche Software eines CP/M-kompatiblen Betriebssystem umfaßt neben dem Dateiverwaltungssystem BDOS mehrere Grundkomponenten im physischen E/A-System BIOS:
- die die Sektorarbeit vorbereitenden BIOS-Rufe **HOME**, **SELDSK**, **SETTRK**, **SETDMA** und **SECTRAN**. Sie dienen zum Laden der spezifischen Schnittstellenparameter vor dem eigentlichen schreibenden oder lesenden Diskettenzugriff. Der zur Zugriffsbeschleunigung über einen logischen Sektorversatz vorgesehene Ruf SECTRAN hat – von älteren Diskettenformaten abgesehen – nur noch eine Dummyfunktion, da als zugriffsbeschleunigende Maßnahmen grundsätzlich die Blockung und gegebenenfalls das Prinzip des physischen Sektorversatzes beim Formatieren der Disketten angewendet werden.
- den Blockungsmodul. Er „sammelt" die vom logischen E/A-System auf der Basis der kleinstmöglichen Verwaltungseinheit von Sektoren a 128 Byte initiierten RD/WR-Rufe und definiert diese in bezug auf eine hypothetische Wirtsdiskette mit einer einheitlichen Sektor-(Blockungs-)größe von 1 KByte. Dadurch erzeugt nicht jeder BDOS-Ruf auch einen physischen Diskettenzugriff. Diese Art der Zugriffsverdichtung ist einer der Wege zur Beschleunigung der Sektorarbeit (ein anderer ist unter anderem die hardwaremäßige Realisierung der CRC-Sicherung durch den FDC). Der Blockungsmodul ist weitgehend hardwareunabhängig. Er kann neben der Blockung auch Aufgaben der automatischen Formaterkennung wahrnehmen, wie das zum Beispiel bei dem in der DDR auch verbreiteten CP/A der Fall ist.
- die physische Floppy-Schnittstelle FD_REQ (FD-Requestvektor) zwischen Blockungsmodul und physischem Modul.

Die Schnittstelle wird vom Blockungsmodul entsprechend der auszuführenden Diskettenoperation geladen. Neben dem Kommandobyte für den Typ der Operation und der Diskettenadresse werden unter anderem der RD/WR-Puffer für die zu transferierenden Daten und ein Zeiger auf einen die physischen Parameter des aktuellen Laufwerkes beschreibenden Diskparameterblock übergeben. Dabei werden die RD/WR-Aufträge als Einsektoroperation – Blockungs- und Sektorgröße auf der Diskette sind gleich – oder Mehrsektoroperation – Blockungsgröße ist ein mehrfaches des Diskettensektors (z. B. beim 16 × 256- oder 9 × 512-Format) – generiert. Rückgabeparameter ist der Status der Operation (Fehler/kein Fehler).

Fortsetzung siehe Seite 170

Entwurf von Gate-Array-Schaltkreisen
Teil 2

Prof. Dr. Dietmar Müller,
Dr. Eberhard Fügert
Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik

CAD-Systeme für den Gate-Array-Entwurf

Der Ablauf und der Umfang des im Teil 1 /1/ beschriebenen Entwurfs ist ohne den Einsatz von CAD-Systemen nicht zu beherrschen.
An (echten) Gate-Array-Kundensystemen werden in der DDR das System U 5200/U 5300 des VEB Forschungszentrum Mikroelektronik Dresden (ZMD) mit dem CAD-System **ARCHIMEDES** und das ISA-System des VEB Halbleiterwerk Frankfurt (Oder) mit dem ISA-CAD angeboten. Am Beispiel von ARCHIMEDES soll der Entwurfsvorgang weiter erläutert werden. Die Komponenten des CAD-Systems ARCHIMEDES sind im Bild 3 dargestellt. Das System ARCHIMEDES setzt insbesondere für die rechenintensive Plazierung, Trassierung und für die Testdatengenerierung die 32-Bit-SKR-Technik (z. B. K 1 840) voraus. Neben diesen rechenintensiven Prozessen ist jedoch eine Reihe von Entwurfsaufgaben nicht zwingend an die 32-Bit-SKR-Technik gebunden. Deshalb wurde an der TU Karl-Marx-Stadt, Sektion Informationstechnik, das zu ARCHIMEDES quelltext- und kommandokompatible System PC-GAD geschaffen, das 16-Bit-PC-Technik nutzt (z. B. EC 1 834, A 7 150 oder XT-Kompatible). Es gestattet den Entwurf (das heißt Schaltungseingabe und Simulation) von Teilschaltungen im System U 5 200 mit einem Umfang von etwa einem Drittel bis einem Viertel eines Gesamtschaltkreises (der Umfang ist stark von der konkreten Struktur abhängig). Mit diesem System kann insbesondere die kreative Phase des Entwurfes von Teilschaltungen am Arbeitsplatz durchgeführt werden. Die Komponenten, die von PC-GAD teilweise oder vollständig erfüllt werden, sind in Bild 3 entsprechend markiert, damit Gemeinsamkeiten und Unterschiede der Systeme erkennbar werden.

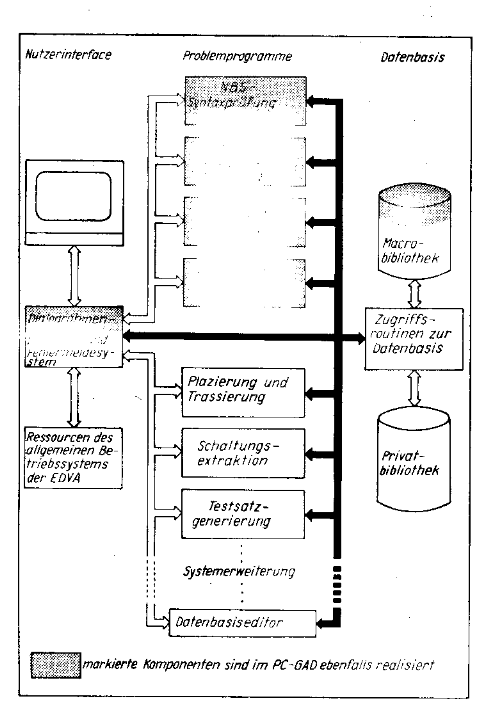


Bild 3 Komponenten von Gate-Array-CAD-Systemen (nach /3/)

Das System ARCHIMEDES unterstützt den Entwurf in der Logikebene bis zum Layout, wobei die Automatisierung des Entwurfes bei den niederen Ebenen steigt. Anders formuliert, der unmittelbare Logikentwurf – das Umsetzen der Sollfunktion in eine entsprechende Struktur – muß manuell oder mit geeigneten (Teil-) Syntheseprogrammen durchgeführt werden. Dieser kreative Entwurfsprozeß wird durch die Aufspaltung der Sollfunktion in Teilfunktionen, die zu Teilschaltungen führen (siehe Teil 1: topdown), erreicht. Das Verhalten dieser Teilschaltungen und später der Gesamtschaltung wird simuliert, und die Struktur wird durch iterative

Schaltungsanpassungen so lange verändert, bis sie die Sollfunktion erfüllt.

Simulationsvorbereitung

Für die Rechnereingabe werden die entworfenen Strukturen sowohl im System ARCHIMEDES als auch im System PC-GAD mit NBS-GA als einem Dialekt der Netzbeschreibungssprache NBS84 /2/ beschrieben. Damit stellt der NBS-Quelltext das Austauschformat zwischen beiden Systemen dar. So können einerseits mit PC-GAD entworfene Teilschaltungen im System ARCHIMEDES als Gesamtschaltung simuliert werden und andererseits bewährte Teilschaltungen als Software-Makros (siehe unten) aus der Makrobibliothek von ARCHIMEDES übernommen werden.
Im Bild 4 ist ein einfaches Beispiel einer NBS-Beschreibung[1] gezeigt. Im Schaltungskopf werden nach den Schlüsselwörtern **S**:, **E**: und **A**: der Schaltungsname und die Namen für die Eingangs- und die Ausgangspotentiale vereinbart. Der folgende Schaltungsrumpf besteht aus den Mehrpolaufrufen mit den vereinbarten Namen und ihren Verbindungen.
Die Möglichkeiten einer hierarchischen Beschreibung sind im Bild 4 erkennbar: Teilschaltungen (im Bild 4 : HA) werden mit ihrem Schaltungsnamen – als Softwaremakro oder Komplexmakro – eingegeben. Diese Teilschaltungen sind im Sinne einer *Dekomposition* in weitere Teilschaltungen aufzulösen. Letztlich müssen jedoch alle Strukturen nur

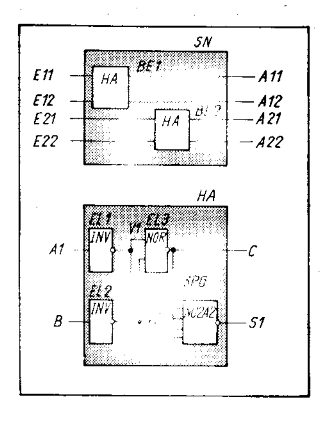


Bild 4 Beispiel einer NBS-Schaltungsbeschreibung

mit den Hardwaremakros aus der Makrobibliothek aufgebaut werden (z. B. &INV1). Diese Hardwaremakros (in NBS-Text durch & gekennzeichnet) sind die vom Hersteller fixierten Funktionselemente, mit denen die Gate-Array-Schaltkreise ausschließlich entworfen werden müssen. Die Hardwaremakros sind umfassend in der Hardwaremakro-Bibliothek (siehe /4/) durch ihr logisches und elektrisches Verhalten und ihr (schematisches) Zellenlayout beschrieben. Diese Parameter werden sowohl vom Simulator als auch von den Plazierungs- und Trassierungsprogrammen benutzt. Im Bild 5 ist ein

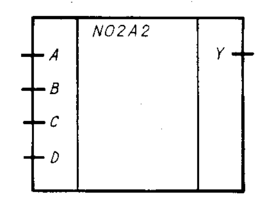


Bild 5 Beispiel einer Elementebeschreibung (Makrobibliothek)

Teil dieser für den Entwerfer zugänglichen Informationen dargestellt, die er benötigt, um beispielsweise die richtige Zuordnung der Ein- und Ausgangspotentialnamen im NBS-Text vornehmen zu können. Die Softwaremakros und die Komplexmakros stellen in diesem Sinne eine Unterstützung des Top-down-Entwurfes dar und bieten die Möglichkeit, durch Rückgriff auf bewährte Teilschaltungen den Entwurfsaufwand zu verringern.
Innerhalb der Simulationsvorbereitung wird durch den **NBS-Compiler** der NBS-Text syntaktisch und semantisch geprüft und der Netzkode erzeugt.
Es folgt die Komponente **Hierarchieauflösung**. Dabei werden alle Teilschaltungen mit Software- oder Komplexmakros so oft aufgelöst, bis nur Hardwaremakros in Mehrpolaufrufen vorhanden sind. Der NBS-Compiler kennt keinen Unterschied zwischen Softwaremakros und Komplexmakros, da beide aufgelöst werden müssen. Für den Entwerfer stellen Softwaremakros geprüfte und bewährte Teilschaltungen dar, die vom Entwickler des Entwurfssystems in die Softwaremakrobibliothek (siehe /5/) aufgenommen wurden. Komplexmakros werden dagegen vom Entwerfer definiert und stellen somit „private" Softwaremakros dar. Durch die Möglichkeit der Nutzung von indizierten Mehrpolaufrufen und von Verbindungsfeldern wird die NBS-Beschreibung sehr effektiv, erfordert jedoch eine Qualifizierung der Mehrpol- und Verbindungsbezeichnungen.
Es schließt sich die Schaltungsprüfung an. Diese Komponente innerhalb von ARCHIMEDES und PC-GAD prüft die Einhaltung von logischen und elektrischen Entwurfsregeln bzw. von Entwurfsrestriktionen. So erfolgt zum Beispiel die Prüfung auf offene Ein- und Ausgänge sowie auf Rückführungen in kombinatorischen Schaltungen, die auf Grund des LSSD-Prinzips (siehe Teil 3) verboten sind.
Bevor der eigentliche Simulationsvorgang beginnen kann, muß eine vollständig aufgelöste und fehlerfreie Schaltung vorliegen, und es müssen Eingangssignalfolgen generiert werden, deren Form – wie oben erläutert – möglichst parallel zum Strukturentwurf festgelegt werden sollte. Diese Signalfolgengenerierung wird durch eine eigene Beschreibungsform (Klammern und Wiederholfaktoren sind möglich) erleichtert. Sie erfolgt bei ARCHIMEDES (für eine schnelle Modifikation) innerhalb und bei PC-GAD (wegen des Zwangs zu speicherökonomischen Lösungen) außerhalb des Simulators.[2]

Simulation

Mit dem **zeitlichen Logiksimulator** können Aussagen gemacht werden zur:
– Erfüllung der Sollfunktion durch die entworfene Iststruktur
– maximal nutzbaren Taktfrequenz
– Lokalisierung kritischer Signallaufzeiten bzw. Signalpfade.

Der Simulator arbeitet mit Verhaltensmodellen der (Hardware-)Makros, die aus einem verzögerungsfreien Logikmodell und aus dem Verzögerungsmodell bestehen. Das Logikmodell beinhaltet neben den üblichen Pegelwerten Low (L) und High (H) einen dritten, mit *unbestimmt* (X) bezeichneten Pegel.
Dieser Signalpegel kennzeichnet einen unbekannten, aber gültigen Pegel (im Unterschied zum verbotenen Spannungsbereich bei TTL-Pegeln). Er wird bei Beginn der Simulation zum Initialisieren und zum Kennzeichnen von *Hazards* benutzt. Bei Ausgangstreibern existiert noch ein vierter, hochimpedanter Zustand Z.
Die Verzögerungsmodelle stellen einen Kompromiß zwischen Laufzeit und Genauigkeit dar und lassen sich zwischen die der üblichen Logik- und der Timingsimulatoren einordnen, wobei die Modelle im PC-GAD wegen der Hauptspeichergröße noch weiter vereinfacht wurden.
Die berechneten Verzögerungszeiten am Hardwareausgang sind abhängig von der Richtung der Signaländerung am Ausgang sowie von der Belastung des Makros durch die Umgebung und durch die restlichen Makroausgänge; dabei wird von Worst-case-Bedingungen ausgegangen. Die berechneten Verzögerungszeiten der Makros innerhalb der entworfenen Schaltung sind mittels Dialogkommandos aufrufbar und damit auswertbar.
Der Simulatorkern basiert auf dem Ereignisprinzip. Dabei werden zu einem gegebenen Zeitpunkt nur *die* Schaltungselemente in die Simulation einbezogen, die zu diesem Zeitpunkt eine Eingangsänderung erfahren. Die Berechnungsreihenfolge für die Logikmodelle und für die Verzögerungsmodelle erfolgt in Abhängigkeit von der Signalausbreitung innerhalb der entworfenen Schaltung. Durch Anwenden des Ereignisprinzips ist die Programmlaufzeit der Schaltungsaktivität direkt proportional. Da im allgemeinen nur ein Drittel der Schaltungselemente gleichzeitig aktiv ist, wird die Programmlaufzeit wesentlich verkürzt.
Wegen der synchronen Arbeitsweise der Gate-Array-Schaltkreise im System U 5200 (siehe Teil 3) sind zwei Simulationsarten für den Entwerfer interessant und werden vom Simulator unterstützt:

① *Simulation des Einschwingvorganges innerhalb einer variablen Taktperiodendauer*
Der Einschwingvorgang umfaßt die Zustandsänderung der Slave-Flipflops (siehe Teil 3), die vollständige Signalausbreitung im kombinatorischen Schaltungsteil und die Signalübernahme durch die FF-Eingangslogik (Set-up-Zeit).
Der Einschwingvorgang ist beendet, wenn in der Gesamtschaltung für die gegebene Signalbelegung der Schaltungseingänge und den aktuellen FF-Zuständen keine Signalwechsel mehr auftreten. Signalwechsel können auch durch Spikes (Hazardfehler) hervorgerufen werden, deren Ausbreitung

[1] Während für die NBS-Eingabe ein Texteditor und ein alphanumerisches Terminal ausreicht, erfordert die komfortablere grafische Eingabe der Logikpläne neben der Gerätetechnik entsprechende Rahmensysteme, zum Beispiel das System MELGET vom VEB Metallurgieelektronik Leipzig für die PC-Technik, oder grafische Editoren, wie TEXgraf vom VEB Textimaelektronik Karl-Marx-Stadt.

[2] Innerhalb des schon erwähnten, grafisch orientierten Systems MELGET steht ein leistungsfähiger Impulseditor zur Verfügung, womit sehr übersichtlich die zeitliche Zuordnung der Eingangsfolgen zueinander erkennbar bzw. eingebbar ist.

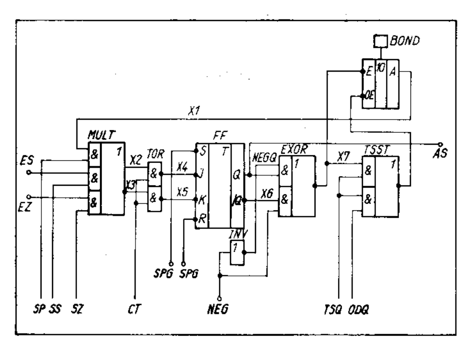


Bild 6 Teilschaltung eines realisierten Gate-Array-Schaltkreises

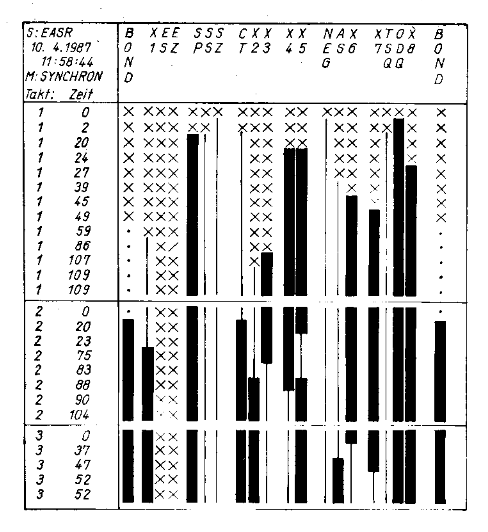


Bild 7 Ausschnitt aus einem Simulationsprotokoll der Teilschaltung aus Bild 6

ebenfalls simuliert wird. Die Zeit der Zustandsänderung der Master-FF wird beim U 5200 nicht betrachtet, da diese allein von der Mastergestaltung und damit vom Systementwickler abhängt.
Die automatische Bestimmung der maximalen Einschwingzeit als ein Maß für die mögliche minimale Taktperiodendauer (Takt-High-Zeit) wird vorgenommen.

② *Simulation des Einschwingvorganges innerhalb einer festen Taktperiodendauer*
Im Unterschied zur ersten Variante erfolgt hier die Simulation bis zur Taktflanke (High-Low-Übergang). Danach wird automatisch eine Kontrollsimulation durchgeführt, um „nachhinkende" Signaländerungen an den Eingängen der Flipflops feststellen zu können. Auf diese Weise sind sich überlappende Einschwingvorgänge simulierbar, wobei die FF-Eingänge bei jeder aktiven Taktflanke auf stationäre Signalwerte geprüft werden.
Während bei der ersten Variante der eingeschwungene Zustand der Schaltung als hinreichende Bedingung zur Sicherung der synchronen Arbeitsweise bezeichnet werden kann, handelt es sich bei der zweiten Variante um die Kontrolle der notwendigen Bedingung (keine Signaländerung an den Eingängen getakteter Elemente). Schaltungen, die intern – abgeleitet aus dem Grundtakt – ein Untertaktsystem besitzen, sind nur in der zweiten Art sinnvoll simulierbar. Davon abgeleitet werden diese Simulationsvarianten als Betriebsart SYNCHRON und Betriebsart UNTERTAKT bezeichnet.
Während der Simulationsvorbereitung wird die Schaltungsstruktur in Listen umgewandelt, wodurch zeitaufwendige Suchprozesse weitgehend vermieden werden. Weiterhin werden nur die Verzögerungsmodelle *der Makros* ausgewertet, deren Logikmodelle einen Funktionswechsel aufweisen. Auch mit dieser Maßnahme wird die Programmlaufzeit gesenkt.
Der dialogorientierte Simulator wird mittels Kommandos gesteuert. Es existieren Kommandos zur:
– Schaltungs- und Simulatorinitialisierung: zum Beispiel Setzen von FF, Potentialen, Takten und Zeiten, Zuordnen von Eingangsfolgen zu den Eingängen, Rücksetzen von Werten
– Simulationssteuerung: zum Beispiel Starten, Unterbrechen, Fortsetzen und Beenden, Wahl der Betriebsart, Setzen von Unterbrechungspunkten
– Ausgabegestaltung der Simulationsergebnisse: zum Beispiel Auswahl anzuzeigender Potentiale und deren Darstellungsform
– Schaltungsanalyse
– Serviceausgabe.
Diese Kommandos und jene zur Signalfolgengenerierung und -verwaltung innerhalb des Systems ARCHIMEDES sind im System PC-GAD in gleicher Weise nutzbar, wodurch eine gleichwertige Nutzeroberfläche entsteht.
Im Bild 6 ist eine Teilschaltung eines Gate-Array-Schaltkreises mit einem Ausschnitt aus einem dazugehörigen Simulationsprotokoll (Bild 7) zu sehen.
Wie im Teil 1 ausgeführt, wird nach der NBS-Eingabe die Komponente Simulator zur Entwurfssimulation genutzt. Diese kreative Phase des Entwurfes kann – neben dem Einsatz des Systems ARCHIMEDES für den Entwurf von Teilschaltungen – vollständig am Arbeitsplatz durch PC-GAD erfolgen. Prinzipiell sollte stets der Entwurf und die Simulation von Teilschaltungen vorgenommen werden, um Aufwand einzusparen und um die Übersichtlichkeit zu erhöhen.
Nach erfolgreichem Abschluß der Gesamtsimulation, das heißt, die Sollfunktion wird durch die entworfene Struktur erfüllt, folgt die Anwendung der CAD-Komponenten für die Plazierung und die Trassierung einschließlich Bondinselplazierung. Die Plazierung und die Trassierung sind sehr rechenzeitintensiv, wobei die Zeit sehr stark von der Ausnutzung der potentiell vorhandenen Logikzellen und der Flipflops abhängt.
Nach der Plazierung und Trassierung wird im System ARCHIMEDES mit dem Simulator die abschließende Bestätigungssimulation durchgeführt. Dabei werden mit den aus dem Layout berechneten Verdrahtungskapazitäten und -widerständen im Rahmen der Modellgenauigkeit die Worst-case-Werte der resultierenden Verzögerungswerte ermittelt. Mit diesen Angaben kann die Frage nach der Erfüllung der Sollparameter (siehe Bild 2 im Teil 1) beantwortet werden.
Das vom Entwerfer unterschriebene Protokoll der Bestätigungssimulation besitzt Dokumentencharakter. Es wird bei der Aufnahme einer Musterpräparation benötigt. Im System ARCHIMEDES werden abschließend Komponenten zum **Erzeugen der Schablonendaten** und zur (Struktur-)**Testdatengenerierung** aktiviert. Damit stehen alle zum Herstellen und Testen eines Gate-Array-Schaltkreises benötigten Daten zur Verfügung, und der Entwurfsvorgang mit dem CAD-System ist abgeschlossen.

Im Teil 3 werden Aussagen zum LSSD-Prinzip, zur Taktgestaltung und zur Schaltungstechnik im System U 5200/U 5300 folgen.

Literatur

/1/ Müller, D.: Entwurf von Gate-Array-Schaltkreisen (Teil 1). Mikroprozessortechnik, Berlin 3 (1989) 3, S. 83
/2/ Autorenkollektiv: Netzbeschreibungssprache NBS-84. Karl-Weierstraße-Institut für Mathematik der AdW der DDR, 1985
/3/ Fischer, W.-I., u. a.: CMOS-Gate-Array-System U 5200. Nachrichtentechnik, Elektronik Berlin 36 (1986) 1, S. 21
/4/ U 5200-Macrozellen-Katalog, VEB Forschungszentrum Mikroelektronik Dresden
/5/ U 5200-Softwaremacro-Katalog, VEB Forschungszentrum Mikroelektronik Dresden

☒ KONTAKT ☎

Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022; Tel. 5 61 31 95

Kleines Lexikon der Mikrorechentechnik

J
wie Jump

Zeichnung: Dahme

Fortsetzung von Seite 168
– den physischen FD-Treiber. Er realisiert letztlich die physische Diskettenarbeit, das heißt das konkrete Lesen und Schreiben von Daten der Diskette. Struktur und Laufzeitverhalten sind voll auf die konkrete Schaltung des FD-Controllers mit dem FDC U 8272 zugeschnitten. Dabei können durch gezielte Maßnahmen und unter Ausnutzung der durch den U 8272 gegebenen funktionellen Möglichkeiten bedeutende Geschwindigkeitssteigerungen bei der Diskettenarbeit und damit verkürzte Programmbearbeitungszeiten erzielt werden.

☒ KONTAKT ☎

Robotron-Meßelektronik „Otto Schön" Dresden, PSF 211, Dresden, 8012

Tools zur BATCH-File-Programmierung

Christian Hanisch, Berlin

Das Konzept der BATCH-Files, die unter MS-DOS neben Programmdateien vom Typ COM und EXE in Form von Prozeduren mit dem Dateityp BAT als gleichberechtigte Kommandos abgearbeitet werden können, erschließt das Gebiet der sogenannten Stapelverarbeitung unter MS-DOS.

Der Kommando-Interpreter COMMAND.COM enthält einige spezielle residente Befehle zur Programmierung von BAT-Dateien. Trotz dieser auf den ersten Blick wenig leistungsfähigen Befehle sind bei Ausschöpfung aller Möglichkeiten dieser Befehle mit nur wenigen spezifischen Zusätzen sehr komfortable Benutzeroberflächen basierend auf einem System von Prozeduren (sprich BAT-Dateien) realisierbar. Zu diesen spezifischen Zusätzen gehören in erster Instanz Programmwerkzeuge zur Dialogabfrage von Alternativen während der Abarbeitung einer BATCH-Datei, zum Setzen von Parameterwerten für globale Variablen des ENVIRONMENTS und zum Übergeben von Zeichenketten in den Tastaturpuffer.

Dialogabfrage

Die Effektivität von BATCH-Dateien erhöht sich wesentlich, wenn während der Abarbeitung durch Bedienereingaben alternative Programmzweige aktiviert werden können. Die Auswertung einfacher J/n-Alternativen kann über den ERRORLEVEL-Befehl erfolgen. Benötigt wird ein Programm, das diesem ERRORLEVEL-Befehl im Register AL einen RETURN-Code von 0..255 übergibt. Neben anderen Lösungen, zum Beispiel dem Programm ASK aus den Norton Utilities, ist das Programm TASTE wegen seiner Einfachheit und leichten Installierbarkeit empfehlenswert. TASTE besteht aus folgenden vier Befehlen:

mov ax,0C08H
int 21H
mov ah, 4CH
int 21H

Diese wenigen Befehle kann man mit DEBUG oder SYMDEB eingeben:
C:\ > DEBUG < TASTE.INP < CR >
Die Datei TASTE.INP besteht aus folgenden Zeilen:

n taste.com < CR >
r cx < CR >
0009 < CR >
a < CR >
mov ax,0C08 < CR >
int 21 < CR >
mov ah, 4C < CR >
int 21 < CR >
< CR >
w < **CR** >
q < CR >

In BATCH-Dateien wird TASTE folgendermaßen angewendet:

echo.
echo RAM-Disk auf Laufwerk „E" (J/n)?
TASTE
:: ASCII-Code 110 = Buchstabe „n"
if errorlevel 111 goto ok
if not errorlevel 110 goto ok
:: Nein-Zweig: NOT Laufwerk „E"
: ok
:: Ja-Zweig: Laufwerk „E"

Abgefragt wird nach dem kleingeschriebenen Buchstaben in der Frage „(J/n)?". Diese Konvention aus /2/ hat sich gut bewährt, weil der Nutzer den Standardzweig einfach durch Drücken der ENTER-Taste aktivieren kann. Falls das Programm TASTE nicht existiert, wird der Standardzweig abgearbeitet.

ENVIRONMENT-Variablen

Das sogenannte ENVIRONMENT ist ein standardmäßig 160 Byte großer RAM-Bereich, in den man unter anderem mit dem SET-Kommando globale Variablen mit ihren Werten eintragen kann, die in BATCH-Dateien gewissermaßen als *COMMON-Variablen* verwendet werden können. Setzt man zum Beispiel mit:

set DRUCKER=EPSON-LX

die Variable DRUCKER auf den Wert *EPSON-LX*, dann kann man in BATCH-Dateien folgendermaßen auf den Variablenwert bezug nehmen:

echo.
echo Drucker ist: % DRUCKER %
echo.

Für % DRUCKER % wird im echo-Befehl der Wert *EPSON-LX* eingesetzt.
Durch:

set DRUCKER=% DRUCKER % 800

wird der Wert der Variablen DRUCKER zu *EPSON-LX800*. Für Anwendungen mit starker Frequentierung des ENVIRONMENTS muß eine Größe von ca. 640 Byte (auf alle Fälle größer als 256) angesetzt werden. Das erreicht man durch folgenden Befehl in der CONFIG.SYS:

shell=\ COMMAND.COM / e : 640 /p

In der Literatur bekanntgewordene Programme /3/ zur dialoggesteuerten interaktiven ENVIRONMENT-Manipulation beschränken sich auf eine Größe des ENVIRONMENTS unter 256 Byte. Mit dem in Bild 1 mitgeteilten Programm MODENV.PAS wird ein ENVIRONMENT beliebiger Größe bedient. Der Quelltext des Programms MODENV.PAS (siehe unten) zeigt zusätzlich folgendes methodische Konzept:
Sowohl die Programmdokumentation als auch der Testrahmen sind Bestandteile des fertigen Programms. Als Konvention könnte gelten: Die Dokumentation eines Programms erhält man durch Aufruf ohne Parameter oder mit dem Parameter „?".
Die Anwendung des MODENV-Programms in BATCH-Dateien ist vielfältig. Als wesentliche Anwendung sei unter anderem das Sichern und Wiederherstellen des aktuellen Verzeichnisses bei Aufruf eines Programmsystems gezeigt:

cd | modenv DIRECT > nul
**cd **
cd | modenv DEVICE > nul

```
{$R-,B-,S+,I+}
PROGRAM MODENV;  { TURBO PASCAL 4.0 }
{ Update und Add einer ENVIRONMENT-Variablen in einem
  ENVIRONMENT beliebiger Größe durch "Tastatureingabe"
  in Batch-Files

  Anwendung: MODENV <SET-VarName> [<any prompt-string>]
  Beispiel : MODENV %1 <DATUM.DAT
  ~~~~~~~~~~ MODENV DATUM Bitte Kontrolldatum eingeben:
             MODENV date TEST Datum eingeben:<DATUM.DAT>PRN:
             CD | MODENV AKTVERZ > nul
  Beachte  : Unter Benutzeroberflächen wie NC, QDOS dgl. funk-
  ~~~~~~~~~~ tioniert die hier beschriebene Technik nicht.
             Unter diesen Kommando-Prozessoren ist ohnehin keine
             unmittelbar wirksam werdende Manipulation des
             Master-ENVIRONMENTS möglich. Zwar könnte dennoch
             durch ein [illegible]es als das hier praktizierte Vor-
             gehen das Master-ENVIRONMENT modifiziert werden. -
             Die Veränderungen würden aber erst nach Verlassen
             der entsprechenden Benutzeroberfläche zum Vorschein
             kommen.                                            }

{*$G4096,P8182,D-
  In TURBO 3.xx müssen zur Redirection diese Compiler-
  Direktiven eingefügt werden.
  Direktive G: Größe des Eingabepuffers bei Redirection
  Direktive P: Größe des Ausgabepuffers bei Redirection  .*}

USES Crt;     (* entfällt in TURBO 3.xx                      *)

CONST line= ══════════════════════════════════════════ ;

TYPE String255=STRING[255];
     StringPTR=^StringEnv;
     StringEnv=ARRAY[1..1] OF CHAR; { ENV-Abbild dynamisch }

VAR   VarName,TempName,Value                        : String255;
      EnvStr                                        : StringPTR;
      i,j,p,VarPos,Curr_Size,Env_Size,Templen,
      Seg_Master_Env, PID                           : INTEGER;

PROCEDURE Get_Env (k:INTEGER; U:StringPTR; VAR J:INTEGER);
{ k  Segment-Adresse des Master-ENVIRONMENTS
  U  zeigt auf ENVIRONMENT-Abbild, das von Get_Env gefüllt wird.
  J  ist die aktuell belegte Länge im ENVIRONMENT.          }

VAR i,m         : INTEGER;
    maybe,skip  : BOOLEAN;
BEGIN
  maybe:=FALSE; skip:=FALSE; i:=1; m:=0;
  REPEAT
    U^[i] := CHAR(MEM[k:m]);
    IF maybe THEN IF U^[i]=#0 THEN skip:=TRUE ELSE maybe:=FALSE
             ELSE IF U^[i]=#0 THEN maybe:=TRUE;
    m:=i; i:=succ(i);
  UNTIL skip;
  J:=m;
END;

PROCEDURE Put_Env (k:INTEGER;U:StringPTR; J:INTEGER);
VAR i,m : INTEGER;
BEGIN
  m:=0;
  FOR i:=1 TO J DO BEGIN MEM[k:m]:=ord(U^[i]); m:=i END;
END;

PROCEDURE showenv(maxlen: INTEGER);
BEGIN
  FOR i:=1 TO maxlen DO BEGIN
    IF (i MOD 67=0) THEN writeln(Output);
    CASE EnvStr^[i] OF
      #0 : Write(Output,chr(221));
      #1..#31: write(Output,'.');
    ELSE write(Output,EnvStr^[i]) END
  END
END;
```

```
PROCEDURE errmessage;
BEGIN
  REWRITE(Output); Writeln(Output); Writeln(Output,
  '*** Error - ENVIRONMENT (',Env_Size,' Byte) voll - "',
  VarName,'" nicht eingetragen !');
  Writeln(Output);
  showenv(Curr_Size);
  Writeln(Output);
  FREEMEM(EnvStr,Env_Size);
  CLOSE(Output);                 HALT (4)
END;

{ ----------------------- M A I N ----------------------- }

BEGIN
  VarName:='NOTHING';   Value:='';  Curr_Size:=0;
  ASSIGN(OUTPUT,''); (* entfällt in TURBO 3.xx                    *)
  rewrite(output);
{ ZUR SACHE:
  Ab 16H des PSP des laufenden Prozesses steht die Segment-
  adresse des PSP des "Eltern-Prozesses". Dies ist i. a.
  COMMAND.COM.                                                     }
  PID:=MEMW[prefixseg:$16]; (* TURBO 3.xx: prefixseg --> CSEG *)

{ Im PSP von COMMAND.COM findet sich nun ab 2CH die Segment-
  adresse des Master-ENVIRONMENTS.                                 }
  Seg_Master_Env:=MEMW[PID:$2C];

{ Bei älteren DOS-Versionen (<3.3) kommt man anders zum Zug:
  Hier erhält man das Segment des Master-ENVIRONMENTS aus dem
  16-Byte-Bereich (MCB = Memory Control Block) vor dem Code-
  Segment des PSP von COMMAND.COM.                                 }
  IF Seg_Master_Env=0 THEN Seg_Master_Env:=PID+MEM[PID-1:3]+1;

{ Die Umgebungslänge wird aus dem 16-Byte-Bereich (MCB des
  Master-ENVIRONMENTS) vor dem Master-ENVIRONMENT abgeholt:       }
  Env_Size:=MEM[Seg_Master_Env-1:3]*16;

  IF ParamCount=0 THEN BEGIN
    Writeln(Output); Writeln(Output,line+' MODENV.EXE =====');
    Writeln(Output,'*** Fatal Error - Parameter missing !!!');
    Writeln(Output);
    Writeln(Output,'A>MODENV <VarName für "SET '+
    '<VarName>=<Wert>"> [<prompt-string>]<CR>');
    Writeln(Output,'           [Testvariante: A>MODENV <VarName>'+
    ' TEST [<any prompt>]<CR>]');
    Writeln(Output);
    Writeln(Output,'errorlevel=255 ---> Fatal Error (siehe oben)');
    Writeln(Output,'errorlevel=  4 ---> ENVIRONMENT voll');
    Writeln(Output,'errorlevel=  0 ---> Ok.');
    Writeln(Output,'Beispiel:               CD | MODENV VERZ  >nul');
    Writeln(Output,'========                if errorlevel 4 echo'+
    ' **** F e h l e r  ***** ');
    Writeln(Output,line+'==================');
    HALT(255)
  END;
  GETMEM(EnvStr,Env_Size);
  VarName:=ParamStr(1);
  FOR i:=1 TO Length(VarName) DO VarName[i]:=UpCase(VarName[i]);
  Get_Env(Seg_Master_Env,EnvStr,Curr_Size);
  TempName:=VarName+'=';
{ ---------------------------- Test ---------------------------- }
  IF (ParamCount>1) AND (COPY(ParamStr(2),1,4)='TEST') THEN BEGIN
    Writeln(Output,line+'==================');
    Writeln(Output,'Maximale ENVIRONMENTlänge (z.B. aus'+
    ' SHELL-Befehl) = ',Env_Size,' Byte');
    Writeln(Output,line+'==================');
    Writeln(Output,'Der Name der SET-Variablen aus ParamStr(1)'+
    'wlautet: "',VarName,'" ');
    Writeln(Output,'Aktuell benutzte Länge des ENVIRONMENTS: ',
    Curr_Size,' Byte');
    Writeln(output);
    showenv(Env_Size);
    Writeln(Output); Writeln(Output);
    Writeln(Output,line+'=================='); Writeln(Output);
  END;
{ ------------------------- Test-Ende ------------------------- }
  Templen:=Length(TempName);  i:=0;
  REPEAT
    i:=succ(i); p:=1; VarPos:=i;
    WHILE (p<=Templen) AND (VarPos>0) DO BEGIN
      IF EnvStr^[i+p-1]<>TempName[p] THEN VarPos:=0;
      p:=succ(p);
    END;
  UNTIL (i>=Curr_Size-Templen-2) OR (VarPos>0);
  IF VarPos>0 THEN BEGIN
    p:=VarPos+Templen;
    WHILE (EnvStr^[p]<>#0) AND (p<Curr_Size) DO p:=succ(p);
    j:=1;
    FOR i:=VarPos TO Curr_Size-Templen-1 DO BEGIN
      EnvStr^[i]:=EnvStr^[p+j];
      j:=succ(j);
    END;
    Curr_Size:=Curr_Size-(p-VarPos+1)
  END;
  IF Curr_Size+Templen>Env_Size-3 THEN errmessage;
  FOR i:=2 TO ParamCount DO write(Output,ParamStr(i),' ');

  CLOSE(Output);
  ASSIGN(INPUT,''); (* entfällt in TURBO 3.xx                      *)
  RESET(Input);    Read(Input,Value);     CLOSE(Input);

  IF Length(Value)>0 THEN BEGIN
    j:=1;
    FOR i:=Curr_Size TO (Curr_Size+Templen) DO BEGIN
      EnvStr^[i]:=TempName[j];
      j:=succ(j);
    END;
    Curr_Size:=Curr_Size+Templen-1;
    IF (Curr_Size+Length(Value)+2)>Env_Size THEN errmessage;
    j:=1;
    FOR i:=Curr_Size TO Curr_Size+Length(Value) DO BEGIN
      EnvStr^[i+1]:=Value[j];
      j:=succ(j);
    END;
    Curr_Size:=Curr_Size+Length(Value);
  END;
  EnvStr^[Curr_Size+1]:=#0;
  EnvStr^[Curr_Size+2]:=#0;

  Put_Env (Seg_Master_Env,EnvStr,Curr_Size+2);
  FREEMEM(EnvStr,Env_Size);
  halt(0)
END.
```

```
PROGRAM KEY; { TURBO PASCAL 4.0 }
{ Aufruf in BAT-Dateien:
    ...
  key y
  turbo-87
    ...
  KEY [F10]^MSKGO^M
  ::(Norton Commander raus, SideKick rein )
  echo off
  echo TURBO PASCAL mit Datei "TEST.BAT"
  key eTEST.BAT^M
  TURBO
  echo Norton-Commander verlassen -
  echo TURBO PASCAL laden, Umschalten auf Lw. E:
  key [f10]yE:^Mturbo^M                                           }

USES crt;  (* entfällt in TURBO 3.xx                             *)

CONST line='================================================';

VAR p,BufferStr,Tmp         : STRING[255];
    Selector,j,code         : INTEGER;

PROCEDURE keybuffer (VAR keys);
{ Schreibt eine Zeichenkette (max. 15 Zeichen) in den
  Tastaturpuffer, dabei belegt ein Zeichen 2 Byte, z. B.
  <F1>  -  00H 3BH
  'A'   -  41H 00H                                                  }
CONST segm=$40; {BIOS-Data Segment}
      offs=$1E; {Keyboard buffer, BIOS stores keystrokes here}
VAR head  : INTEGER ABSOLUTE segm:$1A;{keyboard buffer "head"}
    tail  : INTEGER ABSOLUTE segm:$1C;{keyboard buffer "tail"}
    s     : STRING [255] ABSOLUTE keys;
    j     : INTEGER;

{ Prozeduren CLI und STI so nur in TURBO 4.0     }
PROCEDURE CLI; INLINE($FA); { disable interrupt }
PROCEDURE STI; INLINE($FB); { enable interrupt  }

BEGIN
  head:=offs;  tail:=offs;
  CLI; (* in TURBO 3.xx nur: INLINE($FA);                        *)
  FOR j:=1 TO Length(s) DO BEGIN
    mem[segm:tail]:=ord(s[j]); tail:=succ(tail);
  END;
  STI; (* in TURBO 3.xx nur: INLINE($FB);                        *)
END;

PROCEDURE message;
BEGIN
Writeln(#7#7#7#7); lowvideo;
Writeln(line+' KEY.EXE ========'); highvideo;
Writeln('*** Fatal Error - Parameter unkorrekt ODER'+
' Tastaturpuffer voll !'); lowvideo; Writeln;
Writeln('           KEY   <max. 15 Zeichen für Tastaturpuffer>');
Writeln; Writeln(line+' Ver 2.0 ========');
highvideo; Writeln('Hinweis:'); lowvideo;
writeln('Abschluß "<CR>" als "^M",  signifikantes "^" als "^^"');
Writeln('<CTRL>x als ^x eingeben ---> belegt 1 Zeichen'+
' im Tastaturpuffer.');
Writeln('Funktionstasten F1 .. F12  als   [F1] .. [F12]'+
'  eingeben !!!');
writeln('(Eine Funktionstaste belegt 1 Zeichen im Tastaturpuffer.)');
writeln; highvideo; Write('Beispiel:'); lowvideo;
Writeln(' KEY ye%1^M');
Writeln('            TURBO     ');
Writeln('            KEY [F10]ySK^M');
Writeln('            (Letzteres bedeutet: Norton Commander raus,');
writeln('                                 SideKick rein )');
writeln(line+'=================='); halt(255)
END;

BEGIN
  IF ParamCount=0 THEN Message;
  p:=ParamStr(1); BufferStr:='';
  FOR j:=2 TO ParamCount DO p:=p+' '+ParamStr(j);
  FOR j:=1 TO length(p) DO p[j]:=upcase(p[j]);
  WHILE Length(p)>0 DO
    CASE p[1] OF
    '^': IF length(p)>1 THEN
           IF p[2] IN [#64..#93,#95] THEN BEGIN
             BufferStr:=BufferStr+CHR(ord(p[2])-64)+#0;
             delete(p,1,2)                     END
                                          ELSE
             IF p[2]=#94 THEN BEGIN
               delete(p,1,2);
               IF length(p)>0 THEN
                 IF p[1]=#94 THEN BEGIN
                   BufferStr:=BufferStr+#30#0;
                   delete(p,1,1)  END
                              ELSE BufferStr:=BufferStr+#94#0
                               ELSE BufferStr:=BufferStr+#94
                               END ELSE message
                          ELSE message;
       '[': BEGIN
              Tmp:=copy(p,3,pos(']',p)-3);
              Val(Tmp,Selector,code);
              IF code<>0 THEN message;
              IF (Selector>0) AND (Selector<13) THEN
                 BufferStr:=BufferStr+#0+chr(58+Selector)
                                                  ELSE message;
              delete(p,1,pos(']',p));
            END
      ELSE BufferStr:=BufferStr+p[1]+#0; delete(p,1,1);
    END;
  IF Length(BufferStr) > 30 THEN message;
  keybuffer(BufferStr)
END.
```

Bild 2 KEY.PAS zur Programmierung des Tastatur-Puffers

◀ Bild 1 MODENV.PAS zur interaktiven Manipulation von ENVIRONMENT-Variablen

```
:: DBASE.BAT   -  Aufruf von dBASE III Plus
@echo off
cls
echo
echo                    ^[[1;31;47m╔══════════════════════════╗^[[0m
echo                    ^[[1;31;47m║    d B A S E  III  Plus   ║^[[0m
echo                    ^[[1;31;47m║ Harddisk-Management-System║^[[0m
echo                    ^[[1;31;47m╚══════════════════════════╝^[[0m
echo.
echo.
cd ¦ modenv DIRECT >nul
cd \
cd ¦ modenv DEVICE >nul
subst K: /D > nul
subst K: %DIRECT%
echo  Beachte: Laufwerk K: SUBSTituiert das Verzeichnis %DIRECT%
echo           Keine korrekte Abarbeitung unter NC, QDOS usw. !
echo.
:: Aktuelles Verzeichnis auf  \SYS1\CL\DBASE einstellen:
cd \SYS1\CL\DBASE
KEY ^mset defa to K^M
dbase
subst K:/D
%DEVICE%
cd %DIRECT%
set DEVICE=
set DIRECT=
DIR /W
```

Bild 3 Anwendungsbeispiel für MODENV und KEY in einem dBase-Batch

cd \ sys1 \ cl \ nu ni
% DEVICE %
% DIRECT %
set DEVICE =
set DIRECT =

Mit dem ersten Befehl wird mittels MODENV das eingestellte aktuelle Verzeichnis in die ENVIRONMENT-Variable DIRECT gesichert. Nach Zurückschalten auf das ROOT-Verzeichnis wird das aktuelle Laufwerk in der Variablen DEVICE gesichert. Danach kann auf Laufwerk und Verzeichnis der gewünschten Software umgeschaltet werden. Nach Abarbeitung des Programms werden die in DEVICE und DIRECT gesicherten Werte wiederhergestellt, das ENVIRONMENT *gesäubert* und die BATCH-Datei verlassen. Der Nutzer steht wieder in seinem ursprünglichen Verzeichnis.
Eine andere Anwendung von MODENV wäre zum Beispiel in einer LOGON-Prozedur das Erfragen der USER-Nummer_i, die für die Zuweisung des Systems \ SYSi benötigt wird.

echo.
set LW = C:
modenv USR USER-Nr. (1 .. 15) eingeben:
echo.
**set op = c: \ sys1 \ proclib ; c: \ sys1 \ util ; **
path=% lw % \ sys % USR % \ proclib;
% op %

Die Variable LW kann fest voreingestellt sein, man kann sie aber auch mit MODENV erfragen:

modenv LW Arbeitslaufwerk (c / d):
set LW=% LW %

Tastaturpuffer

Der Tastaturpuffer ist ein Pufferbereich zur Aufnahme von maximal 15 Zeichen, die bei Eingabe von der Tastatur dort zwischengespeichert werden. Der Überlauf des Tastaturpuffers wird akustisch angezeigt.
Falls man von einem Programm aus eine Zeichenkette in diesen Tastaturpuffer schreibt, wird nach Verlassen des Programms der Inhalt das Tastaturpuffers automatisch abgearbeitet, und der Computer reagiert wie von Geisterhand gesteuert.
Mit dem im Bild 2 gezeigten Programm KEY.PAS wird eine in der Aufrufzeile mitgeteilte maximal 15 Zeichen lange Zeichenkette, die auch die Funktionstasten F1 bis F12 als [F1] .. [F12] mit umfassen und CTRL-Steuerzeichen (zum Beispiel < CR > als ^M enthalten kann, in den Tastaturpuffer geschrieben.

Als überzeugende Anwendung sei der Aufruf von Turbo-Pascal Version 3.xx mit automatischer Einbindung der Message-Datei und sofortigem Verzweigen zum Editieren der Datei DEMO.PAS, die auch als Parameter %1 übergeben werden könnte, gezeigt:

echo.
KEY TURBO^MyeDEMO^M

Solche Anwendungen erleichtern die Arbeit wesentlich. Darüber hinaus ist aber noch der Einsatz von KEY als [letzter] Befehl in BATCH-Dateien sinnvoll.
BATCH-Dateien benötigen im RAM einen gewissen Bereich, der solange belegt bleibt, wie die betreffende BATCH-Datei noch aktiv ist. Das kann unter Umständen die gesamte Dauer einer Computersitzung sein, wenn zum Beispiel der Norton-Commander-Aufruf als letzter Befehl in der AUTOEXEC.BAT steht. Wenn später noch weitere BATCH-Dateien auf ähnliche Art benutzt werden, kann es bei Knappheit an RAM-Speicher zu Komplikationen kommen. Gelöst wird das Problem dadurch, daß man eine BATCH-Datei verläßt, in der als [letzter] Befehl zum Beispiel:

KEY ncsmall^M

steht. Dadurch wird die BATCH-Datei vollständig abgeschlossen. Im Tastaturpuffer steht – vom Programm KEY noch aus der BATCH-Datei heraus dort hingebracht – die Zeichenkette: NCSMALL < CR > in *Hab-acht-Stellung*, um nach Verlassen der BATCH-Datei sofort abgearbeitet zu werden.

Literatur
/1/ Bach, M.: Batch-Dateien mit mehr Komfort. mc (1988) 2, S. 82
/2/ Hanisch, Ch.: Wenig beachtete Details bei der formellen Dialoggestaltung von PC-Programmen. MP 2 (1988) 4, S. 98
/3/ Jäger, J.: MS-DOS-Umgebung und Batch-Variable. DOS EXTRA 3/88, S. 18
/4/ Leibrock, G.: Programmierung des Tastaturpuffers unter Turbo-Pascal. DOS (1988) 4, S. 144
/5/ Lindner, F.: Filterprogramme und Pipes unter MS-DOS. MP 2 (1988) 9, S. 263
/6/ Mandrella, P.: Tastatureingaben gespart. DOS EXTRA 4/88, S. 18
/7/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. Franzis-Verlag, München, 1987.
/8/ Zietlow, M.: Programmieren wie von Zauberhand. DOS EXTRA 4/88, S. 16

Softwareansteuerung von Schrittmotoren

Uwe Grunewald
Forschungszentrum des Kombinates VEB Carl Zeiss JENA

Uwe Grunewald (27) legte sein Physikdiplom 1986 an der Friedrich-Schiller-Universität Jena ab und ist zur Zeit als wissenschaftlich-technischer Mitarbeiter im Forschungszentrum des Kombinates Carl Zeiss JENA auf dem Gebiet der Optik-Prüftechnik tätig.

Mit der in diesem Beitrag beschriebenen Softwarelösung ist die Ansteuerung von Schrittantrieben, die die Kommutatorsignale für die Schrittmotoren selbst erzeugen, nach beliebigen Hochlauf- und Bremskurven möglich.
Eine solche Ansteuerung, das heißt das Erzeugen einer Folge kurzer Impulse mit bestimmtem zeitlichen Verlauf der Impulsabstände ist ein Echtzeitproblem, zu dessen Lösung man zumeist auf die Unterstützung autonom arbeitender Elektronik wie Mikrorechner mit CTC oder Einchipmikrorechner zurückgreift. Ist jedoch in einer bestimmten Programmphase die Schrittmotoransteuerung die einzige Aufgabe für den Prozessor eines Mikrorechners, sind also in der entsprechenden Zeit keine Interruptquellen zu bedienen und DMA-Zugriffe ausgeschlossen, benötigt man bei dem vorgestellten Konzept lediglich einen Parallelausgabeport an Hardwareunterstützung.

Vorüberlegungen

Als Zeitbasis für die Impulserzeugung dient die Periodendauer des CPU-Taktes T_T. Das im Bild 1 gezeigte U880-Assemblerprogramm berücksichtigt bei jedem Befehl dessen Taktzahl. Die Grundgedanken der Impulserzeugung sind dabei folgende:
Zuerst müssen an Hand der Datenblätter der verwendeten Motoren bzw. durch Experiment unter konkreten Lastbedingungen einige Parameter festgelegt werden.
1. In welcher Zeit t_{abs} soll von der Start/Stop-Frequenz f_{min} ausgehend welche Maximalfrequenz f_{max} erreicht werden?
2. Wieviele Stufungen n des Frequenzbereiches von f_{min} bis f_{max} sollen möglich sein?
Nach (1) berechnet man entsprechend der festgelegten Frequenzstufung alle möglichen Frequenzen. Danach berechnet man, aus wievielen Impulsen ein jeweiliger Frequenzzug bestehen muß, damit ein Zeitinter-

$$f = H\left(\frac{t}{T} - \frac{1}{2\pi}\left(\sin\left(2\pi\frac{t}{T}\right)\right)\right) \qquad (1)$$

```
;*************************************************************
;   S O F T - M O T O R - C O N T R O L L E R
;   (C) U. Grunewald 12/87
;   VEB Kombinat Carl Zeiss JENA
;*************************************************************

  PORT EQU 80H          ;Adresse Ausgabeport
  OFF  EQU 64           ;Datensatzlänge
; Eingangsparameter
; HL = Schrittzahl      (Schrittzähler)
; IY = Tabellenanfang   (Tabellenzeiger)
; IX = Zeiger auf Bremsentscheidung
; D  = Motorauswahlbyte
; E  = (IY) = Startwiederholzahl (Zähler)
;*************************************************************

MOTOR:  LD A,H                                          (4)
        OR L                                            (4)
        RET Z           ;wenn HL=0                      (5)
        LD A,H                                          (4)
        CP (IX)         ;bremsen?                       (19)
        JP NC,HOCH      ;nein                           (10)
        JP BREMS                                        (10)

HOCH:   LD A,(TAB+OFF-1) ;min. Wartekonstante           (7)
        CP (IY)         ;erreicht?                      (19)
        JP Z,WAO        ;ja                             (10)
        LD A,E                                          (4)
        OR A            ;Wiederholung abgelaufen?       (4)
        JP NZ,WA        ;nein                           (10)
        INC IY          ;ja nächste Wartekonstante      (10)
        JP WAR                                          (10)

WAR:    LD E,(IY+OFF)   ;neue Wiederholung laden        (19)
WART:   DEC E                                           (4)
WARTO:  LD B,(IY)                                       (19)
WART1:  DEC B           ;Warteschleife                  (4)
        JP NZ,WART1                                     (10)
        LD A,D                                          (4)
        OUT (PORT),A    ;Impuls L-->H                   (11)
        XOR A                                           (4)
        DEC HL          ;Schrittzahl-1                  (6)
        OUT (PORT),A    ;Impuls H-->L                   (11)
        JP MOTOR                                        (10)

BREMS:  LD A,(TAB)                                      (7)
        CP (IY)                                         (19)
        JP Z,WAO                                        (10)
        LD A,E                                          (4)
        OR A                                            (4)
        JP NZ,WA                                        (10)
        DEC IY                                          (10)
        JP WAR                                          (10)
WAO:    BIT 0,(IX)      ;61 Takte                       (20)
        BIT 0,(IX)                                      (20)
        IN A,(PORT)                                     (11)
        JP WARTO                                        (10)
WA:     BIT 0,(IX)      ;39 Takte                       (20)
        LD A,I                                          (9)
        JP WART                                         (10)
```

Bild 1

```
;*************************************************************
;   M O T O R D A T E N S A T Z
;   646 Hz ..... 6142 Hz mit 64 Stufen in 1,5 Sekunden
;*************************************************************

; 1. Wartekonstanten

TAB:    DEFB 255,254,254,253,251,248,243,237
        DEFB 229,220,210,198,186,174,162,150
        DEFB 138,127,116,106,97,89,82,75
        DEFB 68,63,58,53,49,45,42,39
        DEFB 36,33,31,29,27,25,24,22
        DEFB 21,20,19,18,17,17,16,15
        DEFB 15,14,14,14,13,13,13,13
        DEFB 13,12,12,12,12,12,12,12

; Wiederholzahlen

        DEFB 15,15,15,15,15,15,15,16
        DEFB 16,17,18,19,20,21,23,24
        DEFB 26,28,31,33,36,38,41,44
        DEFB 48,51,55,59,62,66,70,73
        DEFB 78,82,86,90,94,98,101,106
        DEFB 109,112,115,118,122,122,126,130
        DEFB 130,134,134,134,139,139,139,139
        DEFB 139,143,143,143,143,143,143,143
```

Bild 2

vall $t_{INT} = \frac{t_{abs}}{n}$ vergeht.

Realisierungsbeispiel

Der Soft-Controller besteht aus einer äußeren Schleife konstanter Taktzahl, in der sich eine Warteschleife mit programmierbarer Taktzahl befindet.

Sei nun T_G die Laufzeit der äußeren Schleife und Δt die Zeit für den einmaligen Durchlauf der Warteschleife, dann ergibt sich nach (1) die Frequenz im i-ten Zeitintervall zu:

$$f(t_i) = (f_{max} - f_{min}) \left(\frac{t_i}{t_{abs}} - \frac{1}{2\pi} \left(\sin \left(2\pi \frac{t_i}{t_{abs}} \right) \right) \right) + f_{min} \qquad (2)$$

Man berechnet nun die Zahl der Warteschleifendurchläufe b aus:

$$b(t_i) = \frac{\frac{1}{f(t_i)} - t_G}{\Delta t} \qquad (3)$$

Die Zahlen w, die angeben, wieviele Impulse der jeweiligen Frequenz f (t_i) gesendet werden müssen, damit ein Zeitintervall t_{INT} vergeht, ergeben sich aus:

$$w(t_i) = t_{INT} * f(t_i) \qquad (4)$$

Man erhält so 2 Tabellen, aus der die äußere Programmschleife die aktuellen Parameter bezieht. Diese Schleife benötigt für einen Durchlauf durch ausschließliche Verwendung unbedingter Sprünge konstant 232 Takte. Hierdurch wird ein Impuls erzeugt.

Bei einer Taktfrequenz $f_T = 2{,}4576$ MHz entsprechend einer Periodendauer $T_T = 0{,}4069$ μs ergibt sich:

$T_G = 94{,}4\,\mu s$ und $\Delta t = 5{,}7\,\mu s$.

Da die Zahlen b im CPU-Register B gehalten werden, ergibt sich für b = 0FFH $f_{min} = 646$ Hz und für b = 0 $f_{max} = 10593$ Hz.

In der Praxis wird man die erreichbare obere Grenzfrequenz nicht nutzen können. Die untere Grenzfrequenz läßt sich bei Beachtung der Auswirkung auf f_{max} durch den Einbau von Dummy-Befehlen während der Impulserzeugungsphase (Verbreiterung der Impulse) beliebig verringern.

Die im Bild 2 mit ausgedruckten Tabellenwerte bewähren sich bei der Schrittmotoransteuerung an einem Prüfgerät für die Wellenflächenanalyse von Hochleistungsobjektiven. Die Festlegung n = 64 ist ein Kompromiß zwischen Speicherbedarf (das gesamte Programm benötigt ca. 1/4 KByte) und Frequenzstufung.

Bei der Berechnung der Taktzahlen wurde davon ausgegangen, daß bei Speicherzugriffen keine WAIT-Zyklen stattfinden.

Anwendungshinweise

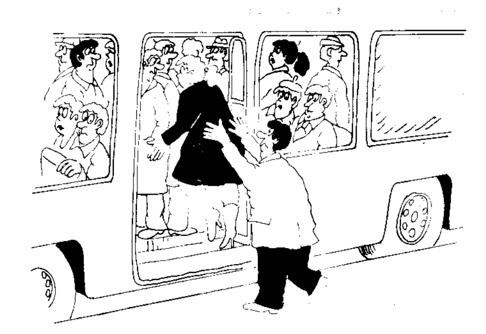
Vor dem Ruf des Soft-Controllers muß in einer vorgelagerten Routine aus der gewünschten Schrittzahl ermittelt werden, nach welcher Schrittzahl wieder gebremst werden muß, um vor dem letzten Schritt sicher die Start/Stop-Frequenz zu erreichen. Der Test des High-Bytes von HL reicht dabei für einfache Anwendungen aus.

Will man 2 Motoren ansteuern, so muß man in einer weiteren Schicht, in der auch das Richtungssignal zu bestimmen ist, das D-Register mit einem entsprechenden Bitmuster laden. Wenn X und Y die Schrittzahlen der beiden Motoren sind, ist man jedoch gezwungen, den Controller über die Bremsentscheidungsroutine zweimal zu rufen. Dadurch wird erst die gemeinsame Schrittzahl und nach Umstellung des D-Registers die Restschrittzahl an den Motor, der die größere Schrittzahl fahren muß, gesendet.

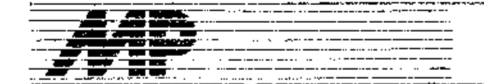
☒ KONTAKT ☎

Forschungszentrum des VEB Kombinat Carl Zeiss JENA, Abt. WOG1 Carl-Zeiss-Str. 1, Jena, 6900; Tel. 83 29 63

Busanpassung



Zeichnung: Steger



Kurs

Turbo-Pascal-Praxis

Teil 1

Manfred Zander, Dresden

In diesem Heft beginnen wir mit einer Kursfolge, die – aufbauend auf dem Kurs Pascal (MP 9 und 11/87 sowie 3, 6, 9 und 11/88) – dem an Turbo-Pascal interessierten Programmierer Hilfsmittel und Werkzeuge für die Beherrschung dieser Sprache in die Hand geben soll, um effektiv programmieren zu können.

Einführung

Turbo-Pascal als eine schnell zu erlernende und doch sehr leistungsfähige Programmiersprache, hat sich in den letzten Jahren in vielen kleinen und mittleren CAD-Zentren unserer Betriebe einen der führenden Plätze erobert. Hier soll keineswegs versucht werden, ein Lehrbuch für diese Sprache vorzulegen, deshalb werden erste eigene Erfahrungen in dieser Sprache vorausgesetzt. Es geht darum, dem Programmierer kleine bewährte Mittel und Methoden vorzustellen, die universell in vielen Programmen genutzt werden können. Dem Anwender soll die Möglichkeit gegeben werden, sich bei der Programmierung voll auf seine Problemlösung zu konzentrieren. Die immer wiederkehrenden „kleinen" Probleme, wie Datenstrukturen, Bildschirmgestaltung, Dialogführung oder Druckerinitialisierung, die bei jeder Anwendung auftreten, belasten leider noch zu oft den Programmierer. Dabei müssen dann Dokumentationen gesichtet werden, Beschreibungen werden gesucht, und jede Lösung wird hundertemal neu gefunden. Dies alles kostet aber nicht nur Zeit. Durch die Nebenbeilösung der kleinen Probleme geraten diese uneffektiv, das wesentliche in den Programmen wird oft verdeckt, die Programme werden zu groß und zu langsam. Der Programmtext wird zu unübersichtlich, die angestrebte gute Strukturierung des Quelltextes leidet, ja es werden sogar Sprungbefehle notwendig. Die Folge all dessen ist natürlich auch eine steigende Fehlerhäufigkeit. Mit dem Vorstellen einiger kleiner Hilfsmittel, die sich bei mir bewährt haben, möchte ich meine Hilfe anbieten.

Von der Gestaltung des Dialoges Programm – Nutzer hängt ganz wesentlich ab, ob eine Software-Lösung später in der Praxis Fuß fassen wird oder ob die Arbeit des Programmierers umsonst war. Daher ist es notwendig, auf diesen Bereich sehr großen Wert zu legen. Programme, die im einfachen Rollmodus mit dem Nutzer kommunizieren, müssen der Vergangenheit angehören. Auch kleinere Lösungen müssen nach außen hin ansprechend und bedienfreundlich gestaltet sein. Selbst bösartige Fehleingaben dürfen nicht zu Programmabstürzen führen. Eine den jeweiligen Erfordernissen entsprechende Menütechnik sollte jedem Nutzer geboten werden. Um durch die Erfüllung dieser Forderungen die Programme nicht unnötig aufzubauschen und deren Lesbarkeit zu erhalten, ist es notwendig, einige Hilfsmittel zu nutzen und die Organisation des Dialoges von der speziellen Lösung zu trennen. Dann können diese Programmteile in einem Include-File versteckt werden und belasten den eigentlichen Programmtext nicht mehr.

Eingabe und Ausgabe auf den Bildschirm

Positioniertes Schreiben

Die wohl am häufigsten immer wieder aufeinanderfolgenden Befehle in Programmen sind gotoxy und write. Ganze Programmpassagen bestehen oft nur aus Kursorpositionierungen und Bildschirmausgaben. Es wird daher die Prozedur writeXY vorgestellt, die in allen Programmen zu Hause sein sollte. Mit ihr wird erst einmal nichts weiter erreicht, als im Programmtext diese zwei Befehle durch einen Prozedur-Aufruf zu ersetzen (Bild 1). Diese Prozedur kann den jeweiligen Bedürfnissen entsprechend modifiziert werden. Als Anregung dazu zwei Beispiele. In der Prozedur writeXYR wird neben der positionierten Ausgabe einer Zahl auf eine bestimmte Bildschirmposition gleich noch eine Standardformatierung dieser realisiert. Somit werden dann alle Zahlen konstant mit drei Nachkommastellen ausgegeben.

Ein Problem bei der Ausgabe von variablen Strings auf den Bildschirm ist das Überschreiben des Bildschirmrandes. Diese Problematik tritt immer dann auf, wenn Zeichenketten ausgegeben werden, die vorher vom Nutzer eingegeben wurden. Um davor sicher zu sein, bewährt sich die zentralisierte Lösung dieses Problems bei der String-Ausgabe, wie es in der Prozedur writeXYS vorgestellt wird.

Positioniertes Lesen

Hier nun einige Hilfen für die Gestaltung von Eingabeanforderungen. Die Befehlskombination gotoxy mit read ist in Programmen häufig anzutreffen. Neben dieser Positionierung der Eingabe auf dem Bildschirm ist aber noch ein weiteres Problem zu lösen. Es soll eine absturzfeste Belegung von Variablen erreicht werden, so daß diesbezügliche Tests im Hauptprogramm entfallen können. Da diese Aufgaben für jede Eingabe gelten, ist wieder eine Zentralisierung der Eingabebehandlung günstig (Bild 2). Als erstes geben wir eine Funktion readXYS für die Eingabe von Strings an, in der natürlich noch keine Fehlerbehandlung notwendig ist.

```
procedure writeXY(x,y:integer;s:string80);
  begin
    gotoxy(x,y);write(s)
  end;


procedure writeXYR(x,y:integer;r:real);
  begin
    gotoxy(x,y);
    write(r:10:3)        (* Formatierte Real-Ausgabe *)
  end;


procedure writeXYS(x,y:integer;s:string80);
  begin
    gotoxy(x,y);
    if length(s)>(80-x) then   (* Verhinderung des *)
      s:=copy(s,1,80-x);       (* Ueberschreibens  *)
    write(s)                   (* Bildschirmrand   *)
  end;
```

Bild 1 Prozeduren zum positionierten Schreiben

```
function readXYS(x,y:integer):string80;
  var s:string80;
  begin                              (* Lesen eines   *)
    gotoxy(x,y);read(s);             (* Strings von   *)
    readXYS:=s                       (* Position x,y  *)
  end;


function readXYR(x,y:integer):real;
  var Io:integer;r:real;s:string80;
  begin
    repeat
      s:=readXYS(x,y);
      while pos(',',s)<>0 do s[pos(',',s)]:='.';
      while pos('+',s)<>0 do delete(s,pos('+',s),1);
      val(s,r,Io)
    until Io=0;
    readXYR:=r
  end;


function readXYI1(x,y:integer):integer;
  var Io,i:integer;s:string80;
  begin
    repeat                           (* Lesen einer    *)
      s:=readXYS(x,y);               (* Integer-Zahl   *)
      val(s,i,Io)                    (* von Pos. x,y   *)
    until Io=0;                      (* bis Test Io    *)
    readXYI1:=i
  end;


function readXYI2(x,y:integer):integer;
  var r:real;
  begin
    repeat                     (* Lesen und Runden      *)
      r:=readXYR(x,y)          (* von Integer-Werten    *)
    until abs(r)<MaxInt;       (* Real-Eingabe erlaubt  *)
    readXYI2:=round(r)         (* bis Integer gueltig   *)
  end;
```

Bild 2 Funktionen zum positionierten Lesen

Für die Eingabe von Real-Werten ist aber eine Behandlung von falschen Eingaben notwendig. Zu schnell geben Nutzer z.B. statt des Punktes ein Komma ein und das Programm stürzt ab. Beim Erstellen einer diesbezüglichen Eingabefunktion ist nun zu entscheiden, ob falsche Eingaben nur zurückgewiesen werden oder ob eine automatische Fehlerkorrektur erfolgen soll. In der vorgestellten Funktion readXYR wird eine solche Korrektur realisiert. Sie liest zuerst einen String, in dem alle Zeichen zugelassen sind. Dann ersetzt sie eingegebene Kommas durch den vorgeschriebenen Punkt und löscht nicht zugelassene Pluszeichen, die ebenfalls zu Programmabstürzen führen würden. Erweist sich die Eingabe danach immer noch als falsch, wird die Eingabe erneut abgefordert.
Für das Einlesen von Integer-Werten bieten sich zwei Varianten an, die jede ihre speziellen Vor- und Nachteile hat. Deren Abwägung hängt von der konkreten Programmieraufgabe ab. In der Funktion readXYi1 wird die Eingabe solange wiederholt, bis die eingegebene Zeichenkette korrekt einer Integerzahl entspricht. Eine automatische Fehlerkorrektur, wie zum Beispiel das Löschen eines möglichen Plus-Zeichens, ist in ihr nicht vorgesehen. In der Funktion readXYi2 ist der Eingabespielraum sehr viel größer abgesteckt. Hier wird die volle Fehlerbehandlung der Funktion readXYR genutzt. Eingaben wie 3,0E10 werden akzeptiert. Da die Funktion readXYR mit Sicherheit eine gültige Realzahl übergibt, genügt ein zusätzlicher Test auf Zahlenbereichsüberschreitung für Integerwerte. Die Funktion readXYi2 übergibt an das Programm den ganzzahlig gerundeten Wert der eingegebenen Zahl.

Kombiniertes Lesen und Schreiben

Durch die Nutzung der bis hier vorgestellten Prozeduren und Funktionen ist das Programm bereits vor einfachen Fehleingaben sicher. Für die Lesbarkeit des Programmtextes ist aber noch ungünstig, daß nicht sofort erkennbar ist, was nun eigentlich an der Position x, y gelesen werden soll, da die Ausgabe auf dem Bildschirm und die Abholung der Eingabe getrennt sind. Die oft in Programmtexten auftretenden Kombinationen der drei Befehle gotoxy, write und read sollten ebenfalls in Funktionen zentralisiert werden. Diese Funktionen realisieren zuerst die positionierte Ausgabe der Eingabeaufforderung auf dem Bildschirm. Direkt dahinter wird dann die absturzfeste Eingabe vom Nutzer entgegengenommen (Bild 3). Diese Aufgabe ist als Beispiel für die Belegung von Real-Variablen in der Funktion holeXYR gelöst.
Benötigt man in einem Programm nicht die volle Vielfalt der bisherigen Funktionen und Prozeduren, kann man die Wirkung der Funktion holeXYR, wie in der Funktion holeXYRsolo gezeigt, auch in sich selbst gewährleisten, ohne auf die Dienste vorangegangener Konstruktionen aufzubauen.
Die Nutzung dieser Funktion könnte dann zum Beispiel so aussehen:

```
voli := holeXYR ('Innenvolumen:',10,5);
```

In dieser komprimierten Befehlszeile ist nun auf einen Blick erkennbar, auf welcher Bildschirmposition welche Variable mit welcher Ausschrift eingelesen wird. Sie beinhaltet funktionell gleichzeitig die Realisierung von Zuweisung, Positionierung, Ausgabe und absturzfester Eingabe. Die einfache Variante dieser Funktion für den Rollmodus, mit dem Namen holeR, soll ebenfalls angegeben werden.
Mit dieser Funktion sieht die Eingabe der Maße eines Kreiszylinders und die Berechnung seines Volumens bereits so einfach aus:

```
Volum:=sqrt ( holeR ( 'Durchmesser' ))
*holeR('Hoehe')*pi/4.0;
```

Eine Zeile, in der einfachste Bedienerführung, Absturzfestigkeit, Lesbarkeit und hoher informativer Gehalt vereinigt sind. Als Veranschaulichung hier das gleiche einmal als einfache Realisierung ohne die hilfreiche Funktion:

```
repeat
  write ( 'Durchmesser : ');
  readln (s);
  val (s,d,lo)
until lo = 0;
repeat
  write ( 'Hoehe : ');
  readln (s);
val (s,h,lo)
until lo = 0;
Volum: = sqrt (d) *h*pi/4.0;
```

Die Vorteile selbst dieser kleinen Hilfsfunktion holeR dürften damit klar auf der Hand liegen, aber wir sind hiermit wieder in den Rollmodus abgeglitten, der für den Nutzer des Programmes unfreundlich ist. Dies würde sich vor allem bei falschen oder fehlerhaften Eingaben bemerkbar machen.

```
/---------------------------------------\
!Eingabe der Masze des Kreiszylinders:  !
!Durchmesser : 10,5                     !
!Durchmesser : 10.5                     !
!Hoehe : ?0                             !
!Hoehe : 20                             !
!                                       !
\---------------------------------------/
```

Bild 4 Bildschirmbild bei einfacher Realisierung

Vertippte sich der Nutzer je einmal bei den beiden Eingaben, ergäbe sich dadurch das in Bild 4 dargestellte Bildschirmaussehen.
Nutzen wir die Funktion holeXYR, sieht die Zeile für die Volumenberechnung des Kreiszylinders so aus:

```
Volum: = sqrt (holeXYR('Durchmesser',10,5))
         *holeXYR('Hoehe',10,6)*pi/
4.0;
```

Die gleichen Vorteile sind erkennbar, aber die Lesbarkeit der Befehlszeile hat schon merklich abgenommen. Dafür haben wir jetzt aber erreicht, daß Fehleingaben das Bildschirmaussehen nicht mehr so stark verunstalten. Bei den gleichen falschen Eingaben hätten wir nach Abarbeitung der Befehlszeile den in Bild 5 dargestellten Bildschirm.

```
/---------------------------------------\
!                                       !
! Eingabe der Masze des Kreiszylinders: !
!                                       !
!                                       !
!        Durchmesser : 10.5             !
!        Hoehe : 20                     !
!                                       !
\---------------------------------------/
```

Bild 5 Bildschirmbild bei Realisierung mit holeXYR

Alle hier vorgestellten Prozeduren und Funktionen sind als Anregung gedacht. Sie können vom jeweiligen Nutzer beliebig modifiziert werden. Denkbar sind für die Lesefunktionen zum Beispiel noch das Verlöschen der Eingabezeile oder auch das Vorgeben von Standard-Eingabewerten. Über die spezielle Form solcher Hilfen wird immer das konkret zu entwickelnde Programm entscheiden. Wesentlich ist die, durch die Anwendung von solchen Hilfsmitteln mögliche, erhebliche Verkürzung von Programmtexten sowie die Zentralisierung der Dialogprobleme. Eine gewissenhafte Beachtung bereits solcher Prinzipien kann sehr viele Schwierigkeiten vermeiden helfen.

Das Auswahlmenü

Auswahlmenüs sind vor allem in dialog-

```
function holeXYR(tXt:string80;x,y:integer):real;
  begin                                (* Auf x,y Schreiben der *)
    writexy(x,y,tXt+' : '):            (* Eingabe-Aufforderung  *)
    holeXYR:=                          (* dahinter Lesen Real   *)
      readXYR(x+length(tXt)+3,y)
  end;


function holeXYRsolo(tXt:string80;x,y:integer):real;
  var s:string80;r:real;Io:integer;
  begin                                (* Auf x,y Schreiben der *)
    repeat                             (* Eingabe-Aufforderung  *)
      gotoxy(x,y);write(tXt+' : ');(* dahinter Lesen der    *)
      read(s);val(s,r,Io)              (* Real-Zahl bis Test Io *)
    until Io=0;
    holeXYR:=r
  end;


function holeR(tXt:string80):real;
  var s:string80;r:real;Io:integer:
  begin
    repeat
      write(tXt+' : ');                (* Eingabe-Aufforderung *)
      readln(s);val(s,r,Io)            (* auf aktuelle BS-Pos. *)
    until Io=0;                        (* Lesen und Zeilen-    *)
    holeR:=r                           (* schaltung bis Io     *)
  end;
```

Bild 3 Funktionen zum kombinierten Lesen und Schreiben

```
const                          (* Tasten-Code :      *)
  Nop         =   #00;         (* Quasileeres Char  *)
  Enter       =   #13;         (* Enter-Taste       *)
  Escape      =   #27;         (* Escape-Taste      *)
  kursortief=     #24;         (* Kursortaste tief  *)
  kursorhoch=     #05;         (* Kursortaste hoch  *)
                  (* Bildschirmsteuerzeichen :  *)
  cleardown =  #$14;           (* Loeschen Bildschirm
                                  ab Kursorposition *)
  kursoron  =  #$82;           (* Kursor ein        *)
  kursoroff =  #$83;           (* Kursor aus        *)
  normvideo =  #$84;           (* normale Darstellung   *)
  invsvideo =  #$85;           (* inverse Darstellung   *)
  intsvideo =  #$86;           (* intensive Darstellung *)
  ivitvideo =  #$87;           (* invers und intensiv   *)
```

Bild 6 Konstanten-Vereinbarungen für PC 1715

orientierten Programmen die hauptsächliche optische Schnittstelle zwischen Programm und Nutzer. Der erste Bildschirmaufbau nach dem Start eines Programms bestimmt den sprichwörtlichen *ersten Eindruck*, aus dem oftmals sofort eine positive oder negative Grundeinstellung erwächst. Und dieser erste Bildschirmaufbau ist in den meisten Fällen das sogenannte Hauptmenü. Es sollte optisch ansprechend, schnell erfaßbar, sowohl leicht als auch komfortabel bedienbar und darüber hinaus auch noch sehr informativ sein. Programmtechnisch steht dem meist die einfache Aufgabe der Belegung einer Character-Variablen gegenüber. Wie wichtig diese Aufgabe aber von international führenden Softwareproduzenten genommen wird, zeigt die Computer- und Softwareentwicklung der letzten Jahre. Da kann der Nutzer Dateien mit einer Maus manipulieren und auf dem Bildschirm in einen Papierkorb fallen lassen. Ausgefeilte Menüs, beispielsweise mit Bildsymbolen, beherrschen den Markt. Die Realisierung dieser nahezu perfekten Lösungen ist aber nicht auf jeder Hardware möglich und vor allen Dingen auch gar nicht nötig. Vorgestellt werden soll daher eine einfache Lösung, die für den PC 1715 erstellt wurde, aber der Erfüllung der oben aufgestellten Forderungen schon sehr nahe kommt. Zur Demonstration der Lösung wird das Hauptmenü eines Spielprogramms genutzt.

Um den Quelltext des Beispielmenüs lesbarer zu gestalten, vereinbaren wir zuerst einige Konstantenbezeichner für die benötigten Tastaturcodes und Bildschirmsteuerzeichen (Bild 6). Diese Definitionen gelten selbstverständlich nur für den PC 1715 und ähnliche Rechner, zum Beispiel die mit einem entsprechenden Bildschirm ausgerüsteten Bürocomputer A 5120 und A 5130.

Entsprechend der programmtechnisch zu lösenden Aufgabe wird das komplette Menü in eine Funktion verpackt, die dem Programm nach seiner Abarbeitung einen gültigen Character-Wert übergibt. Damit wird erst einmal erreicht, daß der für den Aufbau des Bildschirmes zu schreibende Text im Hauptprogramm nicht stört. In dieser Funktion wollen wir nun organisieren, daß der Nutzer die einzelnen Positionen mit einem inversen Balken über Kursortasten anfahren und mit <Enter> auswählen kann.

Die allgemein übliche Möglichkeit der Auswahl mittels Kennbuchstaben bleibt dabei erhalten. Als zusätzlicher Service wird also erreicht, daß der Nutzer bei dieser Menüfunktion nun die Auswahl auf zwei unterschiedliche Arten treffen kann.

Gestaltet man nun den verbleibenden Platz auf dem Bildschirm noch ausreichend informativ und ansprechend, ist diese Ausbaustufe bereits ganz brauchbar und sicher auch für viele Aufgaben ausreichend. Wir wollen aber noch einen Schritt weiter gehen. Um dem Nutzer zu jeder Menüposition zusätzliche Informationen zu bieten, wird organisiert, daß zu jeder Balkenposition rechts neben dem Menü eine invers getastete kleine Tafel aufleuchtet, in der Zusatzinformationen enthalten sind.

Die hier vorgestellte programmtechnische Realisierung einer solchen Menüfunktion basiert auf einer konzentrierten Nutzung von Bildschirmsteuerzeichen. Die Einfassung der Bildschirmtexte, die später intensiv, invers oder invers-intensiv getastet werden sollen, in je zwei Bildschirm-Steuerzeichen normvideo, hat seinen Grund in der Tatsache, daß diese Steuerzeichen im Bildwiederholspeicher je ein Byte Platz beanspruchen. Beachtet man dies nicht, so würde sich der Text auf dem Bildschirm bei den ersten auszuführenden Umtastungen verschieben. Gleichzeitig muß organisiert werden, daß vor Sendung von invsvideo, intsvideo oder ivitvideo an den Bildschirm das Ende des umzutastenden Bereiches bereits mit normvideo abgeschlossen ist, da sonst ein häßliches Aufblitzen des Bildschirmes die Folge wäre. Diese notwendige Einfassung der Bildschirmtexte wird mit der lokalen Funktion norm zentral vorgenommen. Die Erzeugung und die Löschung einer Markierung werden durch die lokalen Prozeduren Markiere und DeMarkiere realisiert. Beide Prozeduren nutzen die für sie globale Variable hoehe, mit der jeweils spezifiziert wird, welche Position zu markieren oder zu demarkieren ist. Die Prozedur Markiere organisiert die Invers- und Intensiv-Tastung einer Menüzeile sowie die Ausgabe der zu dieser Zeile gehörenden Zusatzinformation. Diese Zusatzinformation, die zu jeder Menüposition rechts in der kleinen invers getasteten Tafel erscheint, ist in dieser, zur Funktion Menü lokalen Prozedur, als Konstantenfeld definiert. Jede Tafel kann maximal aus 3 Zeilen, die jeweils 15 Zeichen breit sind, bestehen. Durch die lokale Prozedur DeMarkiere wird diese Markierung einer Menüzeile wieder rückgängig gemacht, indem das Bildschirmsteuerzeichen ivitvideo am Anfang der Menüzeile wieder durch normvideo ersetzt wird. Das Löschen der alten Zusatztafel erfolgt durch einfa-

Bild 7 Funktion Menü

```
function Menue:char;
  const zpunkte='.................................................   ';
  var c:char;i,hoehe:integer;
  function norm(s:string80):string80;
    begin norm:=normvideo+s+normvideo end; (* Einfassen des strings *)
  procedure DeMarkiere;
    begin
      writexy(2,hoehe,normvideo);    (* Zeile wieder in Normaldarstellung *)
      for i:=hoehe to hoehe+2 do     (* alte Tafel wieder normal und leer *)
        writexy(54,i,norm('                 '));
    end;
  procedure Markiere;
    const tafel:array[10..20,1..3]of string[15]=
     ((' Beobachten von',                  (* Definition von 11 moeglichen  *)
       ' auftauchenden ',                  (* Tafeln zu den Menuepositionen *)
       '   Zeichen     '),
      (' Lokalisieren  ','     einer      ',' Bildschirmpos.'),
      (' Erfassen einer',' sechsstelligen','      Zahl     '),
      (' Summieren von ','      fuenf     ','    Ziffern    '),
      ('','',''),('','',''),               (* Leere Tafeln fuer Leerposit.  *)
      ('','',''),('','',''),               (*  Reserve / keine Darstellung  *)
      (' 1 ist einfach ',' 3 ist gut      ',' 9 ist schwer  '),
      ('Zu viele Punkte','    ????????    ',' dann loeschen '),
      ('   Zurueck     ','        zum     ','         System '));
    begin
      writexy(2,hoehe,ivitvideo);    (*  Invers-Intensigve Menuezeile mit  *)
        for i:=hoehe to hoehe+2 do  (*  jeweils aktueller inverser Tafel  *)
          begin
            writexy(54,i,norm(tafel[hoehe,i-hoehe+1]));
            writexy(54,i,invsvideo)
    end   end;
begin
  clrscr;
  write(kursoroff);
  writeln(' K  K    OOO    N   N   ZZZZZ   ![!');
  writeln(' K K    O   O   NN  N      Z     ! ');
  writeln(' KK     O   O   N N N     Z      ! ');
  writeln(' K K    O   O   N  NN    Z       ! ');
  writeln(' K  K    OOO    N   N   ZZZZZ   ]!!');
  writexy(2,10,norm('.Welches.Zeichen....(Aufmerksamkeit).............A  '));
  writexy(2,11,norm('.Wo.war es ......(Aufmerksamkeit und Augenmasz)..B  '));
  writexy(2,12,norm('.Welche Zahl......(Erfassen.und.Merken)..........C  '));
  writexy(2,13,norm('.Rechnen..........(Erfassen.und.Rechnen).........D  '));
  for i:=14 to 17 do writexy(2,i,norm(zpunkte));
  writexy(2,18,norm('.Stellen Schwierigkeitsgrad.......................S  '));
  writexy(2,19,norm('.Loeschen Punkte Stand............................L  '));
  writexy(2,20,norm('.Ende.............................................E  '));
  hoehe:=10;Markiere;

  repeat
    read(kbd,c);c:=upcase(c);
    DeMarkiere;                      (* Loeschen der alten Markierung *)
    case c of
      kursortief: if hoehe<20 then hoehe:=hoehe+1 else hoehe:=10;
      kursorhoch: if hoehe>10 then hoehe:=hoehe-1 else hoehe:=20;
      Enter     : case hoehe of
                    10:c:='A';11:c:='B';12:c:='C';13:c:='D';
                    18:c:='S';19:c:='L';20:c:='E' end
      end;
    Markiere;                        (* Markieren der neuen Position *)
  until c in ['A','B','C','D','S','L','E'];
  Menue:=c;
  write(kursoron)
end;
```

ches Überschreiben mit in normvideo eingefaßten Leerzeichenketten entsprechender Länge. Durch diese nicht zwingend notwendige Aufteilung von Unteraufgaben auf die beiden lokalen Prozeduren und die lokale Funktion ist der Programmtext der Funktion Menü sehr einfach und übersichtlich geworden.

Als erstes wird die Variable hoehe auf 10, das heißt auf die erste Menüposition gestellt und die Markierung durch Aufruf der Prozedur Markiere angewiesen. Die folgende Repeat-Schleife wird nun solange durchlaufen, bis

```
K   K    OOO    N   N   ZZZZZ   III
K  K    O   O   NN  N      Z     I
KK      O   O   N N N     Z      I
K  K    O   O   N  NN    Z       I
K   K    OOO    N   N   ZZZZZ   III

.Welches.Zeichen....(Aufmerksamkeit).............A
.Wo.war es ......(Aufmerksamkeit und Augenmasz)..B  Lokalisieren
.Welche Zahl......(Erfassen.und.Merken)..........C     einer
.Rechnen..........(Erfassen.und.Rechnen).........D  Bildschirmpos
.................................................
.................................................
.................................................
.Stellen Schwierigkeitsgrad......................S
.Loeschen Punkte-Stand...........................L
.Ende............................................E
```

Bild 8 Bildschirmaussehen für das vorgestellte Auswahlmenü

der Nutzer eine gültige Auswahl getroffen hat. Es wird jeweils ein Zeichen von der Tastatur abgefordert. Dieses Zeichen wird in einen Großbuchstaben umgewandelt und die alte Markierung auf dem Bildschirm mit Hilfe der Prozedur DeMarkiere gelöscht. Wurde eine der beiden möglichen Kursortasten betätigt, wird die Variable hoehe auf den neuen Wert eingestellt. Dabei wird eine Ringstruktur der Menüposition realisiert. Hat der Nutzer die Enter-Taste bedient, will er also die zuletzt markierte Option auswählen, wird das Zeichen durch den entsprechenden Auswahlbuchstaben ersetzt (Bild 7).

Die Verbindung zwischen dem Auswahlbalken und dem an das Hauptprogramm zu übergebenden Zeichen wird wieder über die Variable hoehe erreicht, über die nachfolgend auch wieder die Markierung der jeweils angewählten Menüzeile erfolgt. Entspricht nun dieses Zeichen einem möglichen Auswahlbuchstaben, kann die Funktion dieses Zeichen an das Hauptprogramm weitergeben. Wurde aber eine Kursortaste betätigt, wird die nächste Tastenbedienung des Nutzers erwartet. Das von der vorgestellten Menüfunktion erzeugte Bildschirmaussehen kann hier nur schematisch dargestellt werden. Dabei wird die eigentlich invers getastete Menüzeile und die Tafel mit den Zusatzinformationen nur durch Fettdruck gekennzeichnet (Bild 8). Der optische Reiz dieser Lösung besteht in der Dynamik des Menüs bei der Bedienung.

Durch die Einführung der Tafeln zu den einzelnen Menüpositionen bietet unser Menü dem Nutzer ausreichende Informationen zu den einzelnen auszuwählenden Aktionen. Reicht der Platz in den Tafeln nicht aus, kann die Festlegung der Tafelgröße in gewissen Grenzen, die durch die Bildschirmgröße gegeben sind, leicht verändert werden. Diese Ausbaustufe sollte auf kleineren Rechnern nicht mehr überschritten werden. Ein gewisser Service muß zwar realisiert werden, aber man sollte vermeiden, des Guten zu viel zu wollen. Die Anpassung der Funktion an weniger als 11 Menüpositionen ist einfach.

Fenster-Hilfen

Beim Einsatz von Fenstertechniken ist es zum Beispiel möglich, in einem kleinen Bildschirmsegment bei sonst konstantem Bildschirmaussehen eine Vielzahl von Informationen vorbeirollen zu lassen. Dieses Rollen kann auf- oder abwärts erfolgen, je nach der speziellen Zweckmäßigkeit. Es ist auch denkbar, einzelne Fenster über den Bildschirm wandern zu lassen. Allerdings erfolgt bei dem hier vorgestellten Prinzip keine Rettung und Wiederherstellung von anderen überfahrenen Fensterinhalten.

Die vorzustellende Lösung wurde speziell für Compiler entwickelt, die ihrerseits die Window-, Insline- und Delline Befehle nicht realisieren und kann daher durch Nutzung dieser Befehle auf anderen Compilern wesentlich vereinfacht werden. Sie ist sofort auf allen Rechnern lauffähig, die als Prozessor den Z 80 (U 880) enthalten, und den Bildwiederholspeicher von 0F800H bis 0FF7FH aufbauen. Dies ist zum Beispiel bei jedem 8-Bit-PC oder -BC der Fall, sofern man darauf verzichtet, speicherplatzbeanspruchende Bildschirmsteuerzeichen zu verwenden. Um die Lauffähigkeit auf anderen Rechnern zu erreichen, ist aber nur die Änderung jener Komponenten der Lösung notwendig, die Inline-Maschinencode enthalten. Durch die Nutzung dieser Inline-Anweisungen konnte der Programm-Code dieser Lösung angemessen gering gehalten werden. Die Geschwindigkeit der Fensterprozeduren ist ebenfalls auf dieses Verfahren zurückzuführen. Die Prozeduren, die hier vorgestellt werden, können in zwei große Gruppen aufgegliedert werden. Die erste Gruppe ermöglicht die gewünschten Bewegungen auf dem Bildschirm. Dies sind vor allem die Prozeduren inslinedownin und inslineupin. Sie sind die Grundlage, auf der der Großteil der gesamten Lösung basiert. Aufbauend auf diese werden eine Reihe von Prozeduren angegeben, die den Programmierservice schaffen, der für die Beherrschung der Lösung notwendig ist.

Bild 9 Ersatzprozeduren für Insline und Delline

```
procedure inslinedown(i:integer);
  var z:integer;
  begin
    z:=(24-i)*80;
    inline($21/$2f/$ff/           (*LD HL,BS-Ende-80   *)
           $11/$7f/$ff/           (*LD DE,BS-Ende      *)
           $ed/$4b/z/             (*LD BC,(z)          *)
           $ed/$b8);              (*LDDR               *)
    gotoxy(1,i);write(#$16)       (*Loeschen i.Zeile   *)
  end;

procedure inslineup(i:integer);
  var z:integer;
  begin
    z:=(i-1)*80;
    inline($11/$00/$f8/       (*LD DE,BS-Anfang    *)
           $21/$50/$f8/       (*LD HL,Anfang 2.Zeile*)
           $ed/$4b/z/         (*LD BC,(z)          *)
           $ed/$b0);          (*LDIR               *)
    gotoxy(1,i);
    write(#$16);              (*Loeschen i.Zeile   *)
  end;

procedure inslinedownbis(i,u:integer);
  var w,z:integer;
  begin
    z:=(u-i)*80; u:=u*80-1; w:=u-80;
    inline($21/$00/$f8/  (*LD HL,BS-Anfang    *)
           $ED/$4b/u/    (*LD BC,(u)          *)
           $09/          (*ADD HL,BC          *)
           $EB/          (*EX DE,HL           *)
           $21/$00/$f8/  (*LD HL,BS-Anfang    *)
           $ed/$4b/w/    (*LD BC,(w)          *)
           $09/          (*ADD HL,BC          *)
           $ed/$4b/z/    (*LD BC,(z)          *)
           $ed/$b8);     (*LDDR               *)
    gotoxy(1,i);write(#$16)
  end;
```

Grundprozeduren

Als erstes werden in Bild 9 zwei Prozeduren vorgestellt, die als Ersatz für die nicht zur Verfügung stehenden Standardprozeduren Insline und Delline erstellt wurden. Sie sind hier angeführt, um ein schnelleres Verständnis der folgenden physischen Grundprozeduren zu erreichen. In beiden Prozeduren wird auf der durch den Parameter i spezifizierten Zeile des Bildschirmes eine Leerzeile eingefügt. Von dieser Zeile an wird der ursprüngliche Bildschirminhalt verschoben, wobei die letzte bzw. die erste Bildschirmzeile verloren geht. Dieses Wegschieben erfolgt bei der Prozedur inslinedown nach unten und bei der Prozedur inslineup nach oben. Am Anfang beider Prozeduren wird zuerst die Errechnung der Anzahl der zu verschiebenden Bildschirmzeichen vorgenommen, die leicht nachzuvollziehen ist. Da das Verschieben der Zeilen direkt im Bildwiederholspeicher mit Hilfe der Befehle LDDR und LDIR erfolgt, sind die Rechenzeiten dieser Prozeduren minimiert. Das Löschen der eingefügten Bildschirmzeilen wird mit Hilfe eines Bildschirmsteuerzeichens angewiesen, welches eine komplette Zeile löscht und den Kursor an deren Anfang setzt. Durch dieses Vorgehen erhält man eine sehr gute Bilddynamik. Nach der Abarbeitung einer der beiden Prozeduren steht der Kursor am Anfang der eingefügten Zeile.

wird fortgesetzt

Mittel und Methoden der Künstlichen Intelligenz Teil 1

Jörg Schmidt
Technische Universität Karl-Marx-Stadt, Sektion Informatik

Dipl.-Ing. Jörg Schmidt (24) absolvierte von 1982–1987 ein Studium in der Fachrichtung Computertechnik an der Hochschule für Maschinenbau und Elektrotechnik „W. I. Lenin" Sofia/VRB. In seiner Diplomarbeit beschäftigte er sich mit Problemen der Mikrocomputertechnik. Seit September 1987 ist er als wissenschaftlicher Assistent an der Technischen Universität Karl-Marx-Stadt tätig. Gegenwärtiges Arbeitsgebiet sind Möglichkeiten der hardwareseitigen Unterstützung der Künstlichen Intelligenz.

Die moderne Computertechnik durchdringt gegenwärtig alle Bereiche des menschlichen Lebens. Die Computer sind in der Lage, gewaltige Mengen von Daten zu verarbeiten, aber nicht nur Daten schlechthin. Immer mehr geht es darum, den Computern einen gewissen Grad von intelligentem Verhalten aufzuprägen; die Fähigkeit also, sich an wechselnde, vor allem unbekannte Bedingungen um sich herum anzupassen. Dazu benötigen sie einen gewissen Umfang an Wissen, eine Wissensbasis, und die Möglichkeit, mit diesem Wissen auf unbekannte Prozesse, wie Anfragen seitens des Menschen an den Computer, reagieren zu können und dabei Erfahrungen sammeln zu können. Der Computer entwickelt sich zu einem wissensverarbeitenden System.
Im folgenden werden einige Modelle dieser wissensverarbeitenden Systeme beleuchtet sowie eine Übersicht über Eigenschaften von Expertensystemen gegeben.

Wissensrepräsentation

Das Wissen über ein bestimmtes Gebiet besteht im allgemeinen aus einer Menge von Aussagen. Diese können wahr oder falsch sein. Aussagen, wie *„Volker und Kerstin sind verheiratet."*, lassen sich schwer in die Datenstrukturen von herkömmlichen Computerprogrammen pressen. Andererseits ist es aber gegenwärtig auch noch nicht möglich, eine Wissensbasis direkt durch Eingabe natürlicher Sätze zu füllen. Es müssen daher gewisse Schnittstellen zwischen Mensch und Computer bestehen.

Semantische Netze
Grafische Wissensdarstellung

Semantische Netze beruhen auf der alten und sehr einfachen Idee, daß das Gedächtnis – also ein Wissensspeicher – durch Assoziationen zwischen einzelnen Begriffen gebildet wird. Diese Ideen lassen sich bis Aristoteles (384–322 v.u.Z.) zurückverfolgen.
Semantische Netze sind Strukturen, die aus Knotenpunkten und Kanten bestehen. Jeder dieser Knotenpunkte stellt einen Begriff dar. Die Kanten sind gerichtet und widerspiegeln eine Beziehung zwischen den beiden Begriffen, die durch sie verbunden werden. So stellt

ehefrau-von
VOLKER ----------> KERSTIN

eine Beziehung zwischen den beiden Begriffen VOLKER und KERSTIN dar, und zwar, *die Ehefrau von Volker ist Kerstin.* Das ist somit eine Möglichkeit, den eingangs erwähnten Beispielsatz mit einem semantischen Netz zu beschreiben.
Bild 1 zeigt eine weitere Variante, diesen Satz mit Hilfe von semantischen Netzen zu beschreiben. Es ist E1 ein Element aus einer Menge von EHELICHEN-BEZIEHUNGEN. Dann bildet VOLKER den männlichen Teil von E1 und KERSTIN den weiblichen Teil, kurz Mann und Frau. EL stellt die Elementbeziehung dar. Hier wird bereits ein Problem deutlich: Beide Netze widerspiegeln denselben Sachverhalt, haben aber eine andere Struktur. Wollen wir das in Bild 1 dargestellte Wissen um Informationen über die Kinder der beiden bereichern, so erhalten wir die etwas umfangreichere Darstellung in Bild 2.
Aus Sicht der formalen Logik können die Strukturen semantischer Netze wie zweistellige Prädikate aus der Prädikatenlogik aufgefaßt werden. Die in Bild 2 dargestellte Struktur läßt sich dann folgendermaßen ausdrücken:

EL (E1, EHELICHE-BEZIEHUNGEN) &
GLEICH [mann (E1), VOLKER] &
GLEICH [frau (E1), KERSTIN] &
TM (ELTERN, MENSCHEN) &
EL (VOLKER, ELTERN) &
EL (KERSTIN, ELTERN) &
TM (KINDER, MENSCHEN) &
EL (KLAUS, KINDER) &
EL (MARIA, KINDER) &
GLEICH [mutter (KLAUS), KERSTIN] &
GLEICH [vater (KLAUS), VOLKER] &
GLEICH [mutter (MARIA), KERSTIN] &
GLEICH [vater (MARIA), VOLKER].

Umgekehrt lassen sich auch prädikatenlogische Ausdrücke problemlos in binäre Prädikate umformen und somit mit Hilfe von semantischen Netzen darstellen.

Logische Operationen in semantischen Netzen

Wie der prädikatenlogische Ausdruck für das in Bild 2 dargestellte semantische Netz zeigt, sind die einzelnen Beziehungen im Netz mit UND-Verbindungen („&") miteinander verknüpft. ODER-Verbindungen (Disjunktionen) werden gesondert hervorgehoben. Die Kanten, die eine solche ODER-Verbindung bilden, werden eingezäunt, und der Zaun erhält die Bezeichnung ODER. Bild 3 zeigt uns ein semantisches Netz für den Satz: *„Kerstin ist mit Volker oder mit Ulf verheiratet."*
Analog läßt sich die Negation ausdrücken.
Um die Existenz von bestimmten Objekten in den Beziehungen auszudrücken, werden Skolem-Funktionen verwendet. Alle in semantischen Netzen auftretenden Variablen, x, entsprechen der Leseart *für alle* x. Bild 4 zeigt dafür ein Beispiel: *„Volker hat überall etwas verloren."*, d.h., für alle Orte x existiert eine Verlustbeziehung v(x) mit einem verlorenen Objekt o(x), das von VOLKER verloren wurde.
Ähnlich wie die ODER-Verknüpfung lassen sich Aussagen der Form *wenn A dann B* (Implikationen) darstellen. Bild 5 zeigt ein Beispiel für den Satz: *„Wenn Kerstin mit Volker verheiratet ist, dann ist Kerstin nicht mit Ulf verheiratet."*

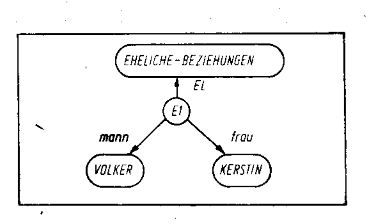


Bild 1 Semantisches Netz für „Volker und Kerstin sind verheiratet."

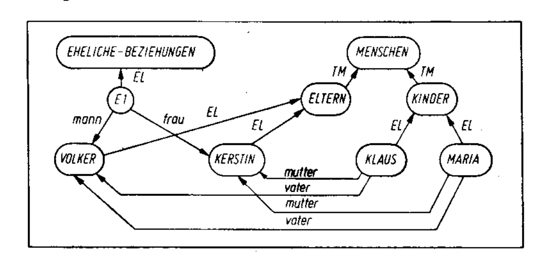


Bild 2 Einbeziehung der Elternbeziehung in das semantische Netz

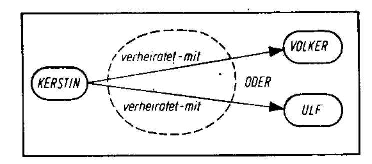


Bild 3 ODER-Verbindung in semantischen Netzen

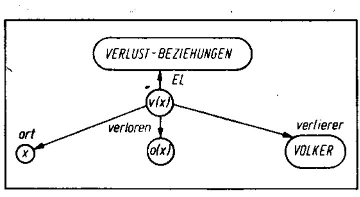


Bild 4 Semantisches Netz mit Skolem-Funktion

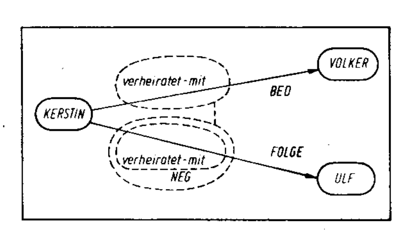


Bild 5 Implikation in semantischen Netzen

Matchen in semantischen Netzen

In semantischen Netzen *matchen* die Zielnetze die Faktennetze, wenn es für jede Kante im Zielnetz eine entsprechende Kante im Faktennetz gibt.
Unter *Matchen* versteht man *etwas passendes suchen.*
Es wurde bereits darauf verwiesen, daß semantische Netze auf prädikatenlogische Formeln zurückgeführt werden können. Folglich können zwei Objekte nur dann gegeneinander *gematcht* werden, wenn eine Vereinheitlichung der entsprechenden prädikatenlogischen Formeln möglich ist.
Ziehen wir noch einmal Bild 1 heran. Wenn wir ausgehend von dem vorhandenen Faktenwissen, daß Volker und Kerstin verheiratet sind, wissen wollen, ob Kerstin die Frau in dieser Ehe ist, so müssen wir die frau-Kante im Zielnetz mit einer frau-Kante im Faktennetz matchen und finden diese Anfrage bestätigt. (Bild 6)

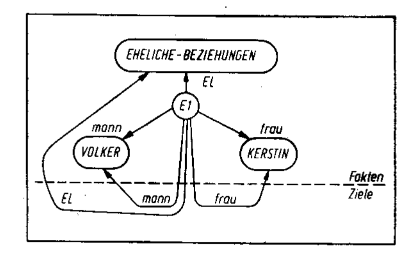


Bild 6 Matchen zweier Netze

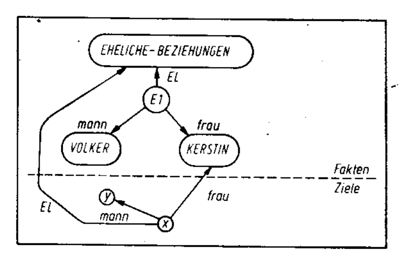


Bild 7 Matchen mit Substitution

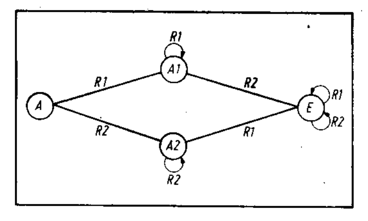


Bild 8 Grafische Darstellung eines kommutativen Produktionssystems

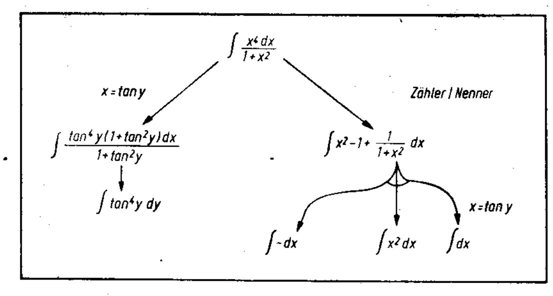


Bild 9 Integralbeispiel

In ähnlicher Weise läßt sich auch nach dem Ehemann von Kerstin fragen. Bild 7 verdeutlicht diesen Sachverhalt. Das Matchen ergibt die Substitution der Variablen x, y als { E1 / x, VOLKER / y }, und wir erhalten VOLKER als Antwort auf unsere Frage.
Probleme entstehen dadurch, daß wir die Möglichkeit haben, Fakten in verschiedener Art und Weise als semantische Netze darzustellen. So würde uns die Darstellung in Bild 7 keine Antwort auf die Frage „*Mit wem ist Kerstin verheiratet?*" geben können. Es sei denn, das System besitzt spezielles Wissen über die Äquivalenz beider Beziehungen.

Produktionssysteme
Grundbegriffe und Beispiele

Produktionssysteme bestehen aus einer globalen Datenbasis, einer Menge von Regeln, die auf die Datenbasis angewandt werden können, um diese zu verändern, und einem System, das die Aufeinanderfolge von verschiedenen oder auch gleichen Regeln steuert (Steuersystem).
Die Anwendung der Regeln auf die Datenbasis verändert diese Datenbasis. Aus einem Anfangszustand wird sie nacheinander in einen Endzustand geführt. Die dabei angewandten Regeln sind ausgehend von der Steuerstrategie bestimmt worden. Sie sind durch ihren Inhalt zu einem bestimmten Zeitpunkt, das heißt auf einen bestimmten Zustand der Datenbasis anwendbar.
In der Literatur wird wegen seiner Einfachheit und Aussagekräftigkeit sehr oft das Beispiel des 8-Zahlen-Puzzles herangezogen. Der Anfangszustand der Datenbasis sei:

2	3	4
1	8	5
7	6	

Dieses Puzzle ist so zu ordnen, daß am Ende

1	2	3
8		4
7	6	5

entsteht.
Als Produktionsregeln seien die vier Bewegungen des Leerfeldes, nach oben, unten links und rechts, möglich.
Die Auswahl der entsprechenden Regel bestimmt die Steuerstrategie. Sie wählt die Regeln aus und merkt sich ihre Reihenfolge. Dabei sind grundsätzlich zwei Strategien möglich. Bei der einen ist eine Regelanwendung unwiderruflich, an den Punkt der Regelanwendung kann nie wieder zurückgekehrt werden; bei der anderen wird die Anwendung einer *Folge von Regeln* erst probiert. Falls sie nicht erfolgversprechend ist, werden andere Regeln angewandt.
Zur letzteren Strategie gehören das Backtracking-Verfahren und die Graphen-Suchverfahren.

Kommutative und zerlegbare Produktionssysteme

Produktionssysteme sind kommutativ, wenn sie folgende Eigenschaften aufweisen:
(1) Wenn die Anwendung von einer Folge von Regeln von der Datenbasis D1 zur Datenbasis D2 führt, so führt auch jedes Vertauschen der Folge der Regeln von D1 zu D2, die Reihenfolge der Regeln ist beliebig.
(2) Jede der auf die Datenbasis D1 anwendbaren Regeln produziert eine Datenbasis, auf die diese Regel wieder angewandt werden kann.
(3) Wenn eine Datenbasis die erzielten Endbedingungen erfüllt, dann führt auch jede Regelanwendung auf diese Datenbasis zu einer Datenbasis, die diesen Bedingungen genügt.
Ein Beispiel für solch ein Produktionssystem ist in Bild 8 veranschaulicht.
Bei zerlegbaren Produktionssystemen (Dekomposition) kann die Datenbasis, auf die eine Regel anzuwenden ist, in Bestandteile zerlegt werden, auf die dann einzelne Regeln angewandt werden. Dadurch entstehen neue Datenbasen, die eventuell wieder entsprechend zerlegt werden können.
Die symbolische Integration soll uns hier als Beispiel dienen. Es ist das Integral

$$\int \frac{x^4 dx}{1 + x^2}$$

zu bestimmen. Bild 9 zeigt den entsprechenden Ablauf. Um das gegebene Integral zu lösen, werden zwei Möglichkeiten ausgewählt aus einer Anzahl von Regeln der Integralrechnung. Die Division Zähler/Nenner führt zu einem spaltbaren Ausdruck. Alle drei Bestandteile müssen weiter verfolgt werden. Sie sind durch eine UND-Verknüpfung miteinander verbunden. Das Ziel, das Ergebnis also, ist erst dann erreicht, wenn alle drei Integrale

$$\int -dx, \int x^2 dx, \int dy$$

gelöst sind. Diese drei Knoten des Graphen in Bild 9 sind sogenannte UND-Knoten. Durch die mögliche Anwendung unterschiedlicher Regeln entstehende Knoten bezeichnen wir als ODER-Knoten. Wir sprechen daher auch von einem UND/ODER-Graphen. Die drei Bestandteile, in die das Integral

$$\int x^2 - 1 + \frac{1}{1 + x^2} \; dx$$

aufgespalten wurde, können unabhängig voneinander gelöst werden; das ermöglicht eine parallele Verarbeitung. Ebenso können aber auch ODER-Knoten parallel weiter verfolgt werden, bis eine Lösung des Problems gefunden wurde.

Frames

Frames sind Mengen von Eigenschaften zu einem bestimmten Objekt oder Ereignis. Ein Frame besitzt einen Identifikator. Das ist der Name des Frames bzw. des Objektes, das beschrieben wird. Diesem Namen folgt eine Reihe von Beschreibungsmerkmalen, die auch als Slots bezeichnet werden.

Beispiel:

```
Name:   Student-1
        Name: Ralph Schuster
        Alter: 22
        Adresse: Karl-Marx-Stadt
```

Ralph Schuster, 22, Karl-Marx-Stadt sind Slotinhalte; in diesem Fall Konstanten. Darüber hinaus können Slotinhalte auch selbst wieder Framenamen sein.
Ein Frame Student kann auch wie folgt aussehen:

```
Name:   Student
        Name: Aggregat (Vorname,Name)
        Alter: Aggregat (Alter)
        Adresse: ADRESSE
```

Dabei deutet ADRESSE auf einen weiteren Frame hin. Die Bezeichnung *Aggregat* verweist darauf, daß bestimmte Objekte (im ersten Fall Vorname und Name) angegeben werden müssen.
Framestrukturen lassen sich sehr einfach in der Programmiersprache Lisp realisieren. So werden sie beispielsweise im Expertensystem MOLGEN (Planung von Experimenten in der Molekulargenetik), das in Lisp implementiert ist, verwendet.

✉ KONTAKT ☎
Technische Universität Karl-Marx-Stadt, Sektion Informatik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 66 85 21

wird fortgesetzt

Wie funktioniert verteilte Arithmetik?

Dr. Ralf Rieken
Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik

Viele Algorithmen zur Realisierung von Digitalfiltern für eindimensionale Signale sowie zur Verarbeitung bildlicher Informationen (Filterung, Transformationskodierung) beinhalten die Berechnung von Produktsummen der Form

$$y = \sum_{i=1}^{N} c_i \cdot x_i \qquad (1)$$

Die programmtechnische Implementierung derartiger Berechnungen bereitet im allgemeinen keine Probleme. Zur Echtzeitverarbeitung sind jedoch äußerst leistungsfähige Prozessoren erforderlich. Modernste universelle Signalprozessoren (DSP 32 C, TMS 320C30, VSP-325) erreichen Spitzenverarbeitungsgeschwindigkeiten im Bereich von 25 bis 37,5 MFLOPS und ermöglichen so die Anwendung von Verfahren der digitalen Signalverarbeitung in vielen Anwendungsbereichen. Die Methode der verteilten Arithmetik ist ein sehr interessantes Beispiel zum Nachweis der Tatsache, daß häufig durch geschickte Modifikation des zu implementierenden Algorithmus elegante Hardwarelösungen gefunden werden. Auf der Basis dieses Verfahrens sind im Zeitalter der ASICs (application specific integrated circuits) Bausteine realisierbar, die bei vergleichsweise niedrigem Integrationsgrad ohne die Verwendung von Multiplizierfeldern die höchstintegrierten Signalprozessoren (bis zu 700000 Transistorfunktionen) bei der Berechnung von Produktsummen übertreffen können.
Das in /1/ beschriebene Verfahren soll kurz vorgestellt werden. Gleichung (1) repräsentiert die Definition des Skalarprodukts der Vektoren $\vec{c}$ und $\vec{x}$. Der Vektor $\vec{c}$ beinhaltet dabei Konstanten (z. B. Filterkoeffizienten). Die Elemente von $\vec{x}$ sind die zu verarbeitenden Abtastwerte. Werden nun die Elemente von $\vec{x}$ als B Bit lange Zweierkomplementzahlen dargestellt, so kann jedes Element x_i wie folgt geschrieben werden:

$$X_i = \sum_{k=0}^{B-1} x_i^{(k)} \cdot 2^{-k} \qquad \text{mit}\, x_i^{(k)} = \begin{cases} 0 \\ 1 \end{cases} \qquad (2)$$

Durch Einsetzen von (2) in (1) erhält man:

$$y = \sum_{k=0}^{B-1} 2^{-k} \sum_{i=1}^{N} c_i \cdot x_i^{(k)} \qquad (3)$$

Die rechte Summe repräsentiert die Produktsumme für die Bitebene k. Da alle c_i a priori bekannt sind, können die für eine Bitebene möglichen 2^N Produktsummen (Zwischensummen) in einer Tabelle abgespeichert werden. Die jeweilige Zwischensumme wird dann durch das Anlegen der k-ten Bitebene von x als Tabellenadresse ausgelesen. Die linke Summe beschreibt die stellenrichtige Addition der Zwischensummen zum Gesamtergebnis. Dazu sind insgesamt B Schritte notwendig.
Die schaltungstechnische Realisierung (Bild 1) erfordert N Schieberegister, einen Tabellenspeicher (RAM) mit 2^N Speicherplätzen sowie einen Addierer mit Registern (z. B. Carry-Save-Addierer /3/). Jedes Schieberegister (SR1 .. SRn) enthält ein Vektorelement x_i. Das niederwertigste Bit von jedem Schieberegister bildet ein Adreßbit für den RAM. Der Addierer summiert die gelesenen Tabellenwerte, indem die Summe des vorhergehenden Schrittes nach Division durch 2 an den rechten Addierereingang gelegt wird. Das niederwertigste Bit jedes Zwischenergebnisses ist bereits ein Ergebnisbit und wird deshalb im Ergebnisregister durch Rechtsverschieben vor dem Überschreiben im nachfolgenden Takt gerettet. Nach B Schritten enthält das Ergebnisregister die Gesamtsumme.
Die Verarbeitungszeit ist unabhängig von der Vektorlänge und der Genauigkeit der Koeffizienten. Für beispielsweise 8 Bit lange Abtastwerte und eine angenommene Verarbeitungszeit von 100 ns je Zwischenschritt liegt das Endergebnis nach 800 ns vor. Bei einer Vektorlänge von 8 entspricht die Berechnung des Skalarprodukts der Ausführung von 8 Multiplikationen und 7 Additionen nach Gleichung (1). Diese einfache Arithmetikeinheit leistet somit bezogen auf einen sequentiellen Prozessor bereits $18{,}75 * 10^6$ arithmetische Operationen je Sekunde, höhere Werte sind möglich.

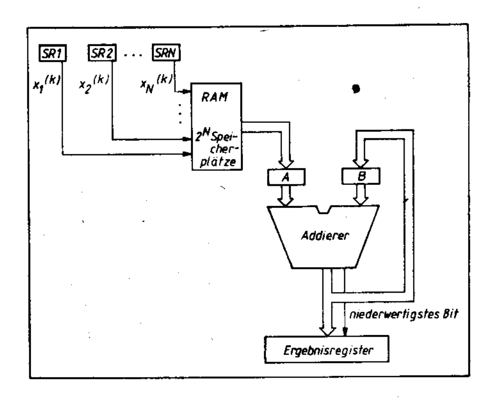


***Bild 1** Struktur einer Verarbeitungseinheit nach dem Prinzip der verteilten Arithmetik*

Der Nachteil des Verfahrens liegt im dem mit der Vektorlänge exponentiell ansteigenden Umfang der Tabelle. Dieses Problem kann allerdings relativ einfach durch die Aufteilung von langen Vektoren auf mehrere unabhängig arbeitende Arithmetikeinheiten gelöst werden (z. B. mehrere Baugruppen für Vektoren mit 8 Elementen). Die Gesamtverarbeitungszeit steigt dabei nur geringfügig durch die notwendige Zusammenfassung der Teilergebnisse an. Die Einführung von *Pipelining* schafft hier jedoch Abhilfe.
Auf der Basis des vorgestellten Verfahrens sind äußerst leistungsfähige Verarbeitungsstrukturen realisierbar. In /2/ wird die Implementierung der rechenzeitaufwendigen diskreten Kosinustransformation auf einem Chip mit nur 12000 Transistoren beschrieben.

Literatur

/1/ Danielson, V.; Ramstad, T. A.: VLSI Implementation of Programmable Digital Filters. Proc. of the Nordic Symposium on VLSI in Computers and Communications, Tampere 1984
/2/ Kronander, T. u. a.: VLSI Implementation of the Discrete Cosine Transform. Proc. of the Nordic Symposium on VLSI in Computers and Communications, Tampere 1984
/3/ Hill, F. J.; Peterson, G. R.: Digital Systems: Hardware Organization and Design. New York: John Wiley & Sons 1978

✉ KONTAKT ☎
TU Karl-Marx-Stadt, Sektion Informationstechnik, PSF 964, Karl-Marx-Stadt, 9010

Diskettenkapazität unter SCP 1700 mit Turbo-Pascal

Winfried Babinsky, Schwerin

Häufig tritt bei Dateiarbeit die Frage auf, wieviel Speicherplatz auf der Diskette noch verfügbar ist. Im folgenden wird eine Turbo-Pascal-Funktion für SCP 1700 beschrieben, die den verfügbaren Speicherplatz als Integerwert in KByte übergibt.
Im Betriebssystem SCP 1700 wird die Belegung der Diskette über die Allocations-Bit-Map verwaltet. Dieser Vektor wird durch das BIOS aus den Directory-Einträgen ermittelt und im Speicher abgelegt. In diesem Vektor bedeutet ein Bit = 1, daß dieser Block belegt ist. Ein Bit = 0 kennzeichnet einen freien Block. Die Blockgröße richtet sich nach dem jeweiligen Diskettenformat und ist im Diskettenparameterblock (DPB) bei Offset 3 durch ein Byte (BLM) definiert. Die Länge des Vektors ist abhängig von der Diskettenkapazität. Diese läßt sich aus dem Produkt aus BLM und der möglichen Anzahl der Blöcke der Platte, welche im DPB bei Offset 5 (DSM) als Wort definiert ist, ermitteln. Dabei ist zu beachten, daß der Wert im DSM die Anzahl der Blöcke minus 1 enthält. Die vorgestellte Funktion wertet diese Parameter selbständig aus, so daß eine umfangreiche Parameterübergabe entfällt. Gleichzeitig ist damit ein universeller Einsatz, für alle unter SCP 1700 lauffähigen Diskettenformate, ohne Änderung der Funktion gewährleistet.
An Hand dieser Funktion soll gleichzeitig der Einsatz von BDOS-Funktionen (BDF) in 16-Bit-Systemen, welche unter Turbo-Pascal eine sehr systemnahe Programmierung ermöglichen, demonstriert werden.
Für die Verwendung von BDOS-Funktionen in 16-Bit-Systemen ist ein Record zu definieren, der die Parameterübergabe realisiert und dabei gleichzeitig als Zwischenspeicher für die Register der CPU beim Interrupt 224, welcher durch die BDOS-Funktion intern aufgerufen wird, dient. Dieser Record ist im Bei-

spiel als CPUREG, die dazugehörige Variable als REG, definiert. Die Auswahl der BDF erfolgt durch das Eintragen der Funktionsnummer in den Lowteil des CX-Registers. Weitere Parameter sind wie in /1/ beschrieben entsprechend der jeweiligen BDF einzutragen. Danach wird mit dem Befehl BDOS (REG) die Ausführung der BDF veranlaßt.

Der Aufruf der Funktion erfolgt mit:
Var:=DISKKAP(LW,LWZ);

Hierbei ist LW das Laufwerk, für das die Speicherkapazität ermittelt werden soll. LW und LWZ sind als CHAR anzugeben.
Der Einsatz dieser Funktion in 8-Bit-Systemen unter SCP ist problemlos möglich. Es ist dabei nur die Veränderung der Adressen, keine Segmentangabe, zu beachten und entsprechend im Quelltext zu ändern.
Dieser Beitrag soll eine Anregung sein, unter Nutzung von BDOS-Funktionen unter Turbo-Pascal systemnah zu programmieren. Für versierte Pascal-Programmierer ist es sicherlich günstiger, diese Funktion in Assembler zu schreiben und sie mit der INLINE- bzw. EXTERNAL-Anweisung in den Quelltext einzubinden.

Literatur
/1/ Steuerprogramm SCPX 1700, VEB Robotron-Projekt Dresden


```
{16.08.88}
FUNCTION DISKKAP(LW,LWZ:CHAR):INTEGER;
TYPE
CPUREG=RECORD
        AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS:INTEGER;
        END;

VAR
REG:CPUREG;
DPBSEG,DPBOFS,ALSEG,ALOFS,DK,I1,I3:INTEGER;

ALN,BG,I2:BYTE;

BEGIN
I2:=0;
WITH REG DO BEGIN
{ KAPPAZITAET FUER WELCHES LW ? }
{ AKTUALISIERUNG DES ALLOC-VEKTORS}
CX:=$000E;          {BDF 14}
CASE LW OF
        'A':DX:=$0000;
        'B':DX:=$0001;
        'C':DX:=$0002;
        'D':DX:=$0003;
        'E':DX:=$0004;
        END;
BDOS(REG);
CX:=$000D;           {BDF 13}
BDOS(REG);
{ ADRESSE ALLOC-VEKTOR ERMITTELN }
   CX:=$001B;        {BDF 31}
   BDOS(REG);
   ALOFS:=BX;
   ALSEG:=ES;
{ ADRESSE DPB ERMITTELN }

   CX:=$001F;
   BDOS(REG);
   DPBSEG:=ES;
   DPBOFS:=BX;
   END;
{ BLOCKGRöSSE UND ANZAHL DER BLöCKE-1 AUS DPB}
{ZUGRIFF AUF SPEICHERPLäTZE MIT TURBO-FKT. MEM BZW. MEMW}
CASE MEM[DPBSEG:DPBOFS+3] OF
                        7:BG:=1;
                       15:BG:=2;
                       31:BG:=5;
                       63:BG:=8;
                      127:BG:=16;
                        END;
{ DISKETTENKAPAZITAET ERMITTELN }
DK:=BG*(MEMW[DPBSEG:DPBOFS+5]+1);

{ LAENGE ALLOC_VEKTOR ERMITTELN }
ALN:=MEMW[DPBSEG:DPBOFS+5] DIV 8;
{ BIT-BELEGUNG ERMITTELN }
I3:=0;
FOR I1:=ALOFS TO ALOFS+ALN DO BEGIN
                     I2:=MEM[ALSEG:I1];
IF (I2 AND 128) = 128 THEN I3:=I3+1;
IF (I2 AND 64) = 64 THEN I3:=I3+1;
IF (I2 AND 32) = 32 THEN I3:=I3+1;
IF (I2 AND 16) = 16 THEN I3:=I3+1;
IF (I2 AND 8) = 8 THEN I3:=I3+1;
IF (I2 AND 4) = 4 THEN I3:=I3+1;
IF (I2 AND 2) = 2 THEN I3:=I3+1;
IF (I2 AND 1) = 1 THEN I3:=I3+1;
                     END;
 DISKKAP:=DK-BG*I3;        {BG*I3 = BELEGTE SPEICHERKAPAZITAT}
{-------------- AKTUELLE LW ZURUECKSTELLEN --------------}
WITH REG DO BEGIN
       CASE LWZ OF
                'A':DX:=$0000;
                'B':DX:=$0001;
                'C':DX:=$0002;
                'D':DX:=$0003;
                'E':DX:=$0004;
                END;
       CX:=$000E;
       BDOS(REG);
       END;
END;
```

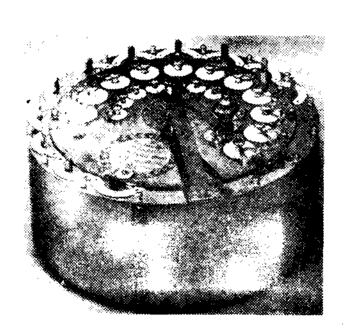
Wegbereiter der Informatik



PHILIPP MATTHÄUS HAHN

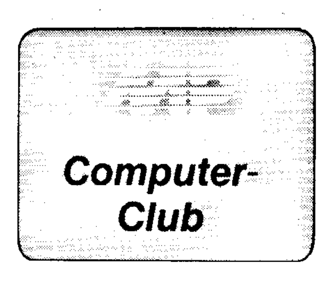
*** 1739 Scharnhausen/Württ.**
+ 1790 Echterdingen/Württ.

Der Schatten eines Nagels hatte P. M. Hahn dazu angeregt, sich in seiner Jugendzeit mit Sonnenuhren und intensiv mit Astronomie zu befassen. Diese praktische Neigung wurde ihm während seines ganzen späteren Lebens zum zweiten Beruf, so daß er sich schließlich als ausgezeichneter Erfinder und Mechaniker einen Namen machte.
Doch bestimmte ihm das Elternhaus den Beruf eines Theologen. Von seinem Vater, der selbst Pfarrer war, wurde er schon frühzeitig auf diese Laufbahn vorbereitet: Bereits mit vier Jahren erhielt er von ihm Unterricht in den Altsprachen Latein, Griechisch und Hebräisch. Während seines entbehrungsreichen Theologie- und Philosophiestudiums 1757–1760 an der Universität Tübingen kamen ihm seine mechanischen Kenntnisse insofern zustatten, als er sich mit der Anfertigung von Sonnenuhren seinen Studienunterhalt mit bestreiten konnte. Nach Übernahme (1764) eines Pfarramtes in einem württembergischen Dorf unterhielt er von nun an in seinem Pfarrhaus ständig eine feinmechanische Werkstatt, in der mehrere Gehilfen und dann auch einige seiner Söhne beschäftigt waren. In dieser Werkstatt baute man von Hahn erfundene oder verbesserte Präzisionsuhren in Taschen- und Großformat, Standuhren mit astronomischen Beiwerken, Sonnenuhren, Planetarien bzw. astronomische Uhren, die das Geschehen im erd- und sonnennahen Raum wiedergaben.
Die schwierigen Berechnungen für all diese Präzisionsgeräte mögen wohl Hahn schließlich veranlaßt haben, sich dem Bau einer Rechenmaschine für alle vier Grundrechenarten zuzuwenden; das 1. Exemplar war 1774 fertiggestellt



(siehe Bild). In dieser Maschine hat Hahn das von Leibniz erfundene Prinzip der Staffelwalze verarbeitet und ein festes Staffelwalzen-Stellwerk eingebaut. Die Betätigung der zentralen Antriebskurbel bewirkt eine serielle Addition der (offenbar 11stelligen) Zahlen des Stellwerkes zum innen gelegenen Summenwerk. Neuartig und konstruktiv günstig durchdacht ist die dazu gewählte runde Trommelform. Hierin folgte Hahn einem Vorschlag des Mathematikers und Mechanikers Jakob Leupold (1674–1727), der das Wissen seiner Zeit auf den Gebieten der praktischen Mechanik und der Rechentechnik in einem 12bändigen Werk („Theatrum Machinarum") zusammengefaßt hat.
Bis zur Gegenwart wird P. M. Hahn – besonders in seiner Heimat – ein hohes ehrendes Andenken bewahrt, wird doch seine mechanische Werkstatt als die Keimzelle der späteren Württembergischen feinmechanischen Industrie angesehen! Er gilt dort als Begründer der Waagenindustrie und der Feinwerktechnik. Seine Maschine wurde die erste industriereife Rechenmaschine, von der noch heute einige Exemplare funktionstüchtig sind.

Dr. Klaus Biener



Commodore-Interface am U 880

Commodore verwendet bei seinen Heimcomputern einen eigenen Peripheriebus, der eine einfache Anschaltung der verschiedenen Datenendgeräte an den Computer erlaubt. Doch was ist zu tun, wenn man einen Drucker mit Commodore-Interface an einen Rechner mit dem Prozessor U 880 anschließen möchte?

Die Commodore-Schnittstelle

Es handelt sich dabei um ein Interface mit serieller Busstruktur, das anders als bei einer V.24-Schnittstelle die Parallelschaltung von bis zu 15 Geräten am Bus erlaubt. Die Peripheriegeräte hören zunächst nur passiv am Bus. Erst durch Befehle vom Controller (Rechner) werden sie aktiviert. Das Interface ist eine vereinfachte Version des Standard-IEEE-488-Busses.

Neben der Masse gibt es die Leitungen ATN (Attention), DATA, CLOCK und bei Bedarf IFC (Interface Clear = Reset) für ein Hardware-Reset.

ATN

Dieses Signal dient der Peripherie zur Unterscheidung zwischen Befehls- und Datenbytes. Low-Pegel bedeutet Übertragung von Befehlen vom Computer zur Peripherie, High-Pegel kennzeichnet Daten.

CLOCK

Der Datensender steuert mit dem CLOCK-Signal die Übernahme der Datenbits durch den Empfänger. Durch die steigende Flanke von CLOCK wird das Datenbit eingelesen.

DATA

Vom Sender wird das Daten- bzw. Befehlsbyte seriell – beginnend mit dem LSB – auf die Datenleitung geschoben.

IFC

Mit IFC = Low wird ein Reset in den Peripheriegeräten ausgelöst.

Die Leitungen ATN und IFC werden grundsätzlich nur vom Computer gesteuert, der somit jeglichen Befehls- und Datentransport am Bus kontrolliert. Der Computer fordert einzelne Peripheriegeräte als Listener (Hörer) oder Talker (Sprecher) auf und übergibt im letzten Fall die Steuerung des Datentransfers zeitweilig dem konkret angesprochenen Gerät, das heißt, die Leitungen CLOCK und DATA werden durch den aktiven Talker kontrolliert. Aus diesem Grund und wegen der noch zu beschreibenden Quittierungszyklen sind beide Leitungen mit Open-Kollektor-Gattern und 1-kΩ-Pull-up-Widerständen zu versehen. Sie werden vom aktiven Gerät auf Low-Potential gezogen /1/.

Hardware

Rechnerseitig werden für die Interface-Steuerung mindestens 4 Bit eines PIO-Kanals benötigt. Es sind dies die Leitungen ATN-OUT, CLOCK-OUT sowie DATA-IN/OUT. Wird das Interface vollständig aufgebaut, kommen noch die Signale CLOCK-IN (z. B. für den Anschluß einer Commodore Floppy-Station) und IFC hinzu. Bild 1 zeigt die Schaltung der Anpaßstufe. (Zwei der vier Gatter des D 103 könnten frei bleiben, wurden jedoch zweckmäßigerweise zum Treiben von ATN und IFC eingesetzt.) Weitere Hardware, zum Beispiel ein CTC, ist nicht notwendig, da die Zeitbedingungen aufgrund der bitsynchronen Übertragung relativ unkritisch sind und über Zählschleifen erfüllt werden können.

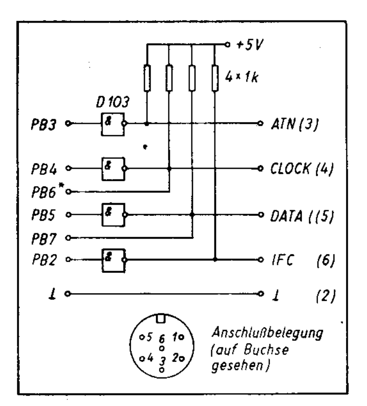


Bild 1 Schaltung der Anpaßstufe für das Commodore-Interface (nur bei Anschluß eines Talkers am Bus notwendig)*

Datenübertragungsdiagramm

Beschränkt man sich auf den Anschluß eines Listeners am Bus, so wird ein Quittierungszyklus vom Empfänger zum Sender praktisch nur nach dem Senden eines vollständigen Bytes benötigt. Des weiteren gibt es noch einen ATN-Bestätigungszyklus (nach dem Schalten von ATN = Low) und einen End-of-Information-Quittierungszyklus, die jedoch beim Anschluß nur eines Gerätes entfallen können. Unter diesen Gesichtspunkten zeigt Bild 2 ein Schema der Abläufe zur Datenübertragung vom Computer zur Peripherie.

Das Aktivieren eines Peripheriegerätes erfolgt durch Aussenden der entsprechenden Primäradresse, optional gefolgt von der Sekundäradresse. Das ist dem Gerät durch ATN = Low vor LOCK = Low (T1) anzuzeigen. (Die Bestätigung erfolgt durch die Peripherie, indem nach T1 die DATA-Leitung für 1 ms Low wird.) Nachdem der Empfänger das letzte Befehlsbyte quittiert hat (T12), wird ATN = High.

T1 Mit CLOCK = Low wird vom Computer die Sendeanforderung gestellt.

T2 Der Empfänger quittiert die Anforderung mit DATA = Low (T2-T1 ≦ 1 ms).

Bild 2 Diagramm zur Interface-Steuerung

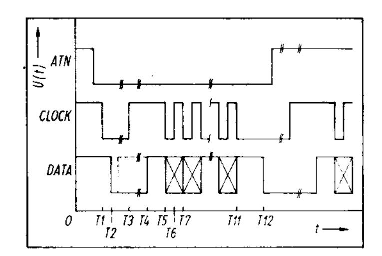


Bild 3

```
**********************************************
*      DRUCKERANSTEUERUNG              14.10.88  *
*      COMMODORE-INTERFACE                      *
**********************************************

DRUCKERINITJALISIERUNG
aktiviert den Printer am Peripherie-Bus
        ORG     8000H
WBOOT   EQU     0A003H
IN0     LD      HL,TAB   ;HL --> Initial.-Tabelle
IN1     LD      A,(HL)
        CP      0FFH     ;Ende-Kennung?
        JP      Z,WBOOT  ;Return zum System
        CALL    MAIN     ;HauptprogrammAufruf
        INC     HL
        JR      IN1

**********************************************
*HAUPTPROGRAMM MIT FUNKTIONSVERTEILER (=Treiber)*
*Datenbyte in <A>                               *
*Steuerzeichen des Druckertreibers              *
* 02H = Leitung ATN LOW                         *
* 03H = Leitung ATN HIGH                        *
* 04H = PIO-INITIALISIERUNG                     *
* 05H = Hard-RESET Printer                      *
**********************************************
PORT    EQU     041H     ;Daten-Adr. PIO-PORT B
MAIN    PUSH    HL
        PUSH    DE
        PUSH    BC
        PUSH    AF
        LD      D,A
        LD      C,PORT
        LD      HL,AZ    ;HL --> AZ
        LD      E,(HL)   ;letzten Ausgabewert holen
        CP      2        ;Funktion ATN LOW ?
        JR      Z,ATL
        CP      3        ;Funktion ATN HIGH?
        JR      Z,ATH
        CP      4        ;Funktion PIO-INIT
        JR      Z,PIO
        CP      5        ;Funktion RESET
        JP      Z,RESET
        JR      FRAM     ;Daten zum Drucker
AUS     LD      (HL),E   ;Ausgabewert sichern
        POP     AF
        POP     BC
        POP     DE
        POP     HL
        RET

ERR     JP      WBOOT    ;Fehler:RETURN zum System
PT ATN LOW
Steuerleitung ATN aktivieren fuer Befehlsausgabe
ATL     SET     3,E      ;ATN LOW
        OUT     (C),E
        SET     4,E      ;TAKT LOW
        OUT     (C),E
        LD      B,63     ;1ms warten
ATL1    IN      A,(C)
        BIT     7,A      ;Zustand DATEN?
        JR      Z,AUS    ;Warten auf DATEN=LOW
        DJNZ    ATL1
        JR      ERR

PT ATN HIGH
ATN desaktivieren
ATH     RES     3,E      ;ATN HIGH
        OUT     (C),E
        JR AUS

PT FRAME
Kern zur Ausgabe von Daten an den Drucker
FRAM    RES     4,E      ;TAKT HIGH
        OUT     (C),E
FRA1    IN      A,(C)
        BIT     7,A      ;Zustand DATEN
        JR      Z,FRA1   ;Warten auf DATEN=HIGH
        LD      B,7      ;40 mikro s warten
FRA2    DJNZ    FRA2
        LD      A,D
        CALL    DOUT     ;DATEN seriell ausgeben
        LD      B,63     ;1 ms warten
FRA3    IN      A,(C)
        BIT     7,A      ;Zustand DATEN
        JR      Z,AUS    ;DATEN LOW
        DJNZ    FRA3
        JR      ERR

PT DATEN SERIELL AUSGEBEN
Kern zur seriellen Ausgabe eines Byte
DOUT    LD      B,8      ;DATEN 8*schieben
DO1     SET     4,E      ;TAKT LOW
        OUT     (C),E
        PUSH    BC
        BIT     0,A      ;DATENBIT testen
        JR      NZ,DO3   ;Bit=1
        CALL    DOL      ;Bit=0
DO2     RRC     A        ;DATEN schieben
        POP     BC
        DJNZ    DO1      ;8 Bit ausgegeben?
        SET     4,E      ;TAKT LOW
        RES     5,E      ;DATEN HIGH
        OUT     (C),E
        RET
DO3     CALL    DOH
        JR      DO2
```

```
DOL     SET     5,E        ;Ausgabe LOW-Bit
DO4     OUT     (C),E
        LD      B,5
DO5     DJNZ    DO5        ;Zeitverzoeg.
        RES     4,E        ;TAKT HIGH
        OUT     (C),E
        LD      B,1
DO6     DJNZ    DO6        ;Zeitverzoeg.
        RET
DOH     RES     5,E        ;Ausgabe HIGH-Bit
        JR      DO4

PT PIO
PIO Initialisierung
PIO     LD      C,PORT+2;SteuerwortAdresse Port B
        LD      E,0CFH     ;Programm.MODE 3
        OUT     (C),E
        LD      E,0C3H     ;BIT EEAAAAEE
        OUT     (C),E
        LD      C,PORT     ;DatenwortAdresse
        OUT     (C),E      ;TreiberAusgaenge HIGH
        JP      AUS

PT RESET
Hardware-RESET
RESET   SET     2,E        ;B2 LOW
        OUT     (C),E
        RES     2,E        ;B2 HIGH
        OUT     (C),E
        LD      B,6        ;6fache
ST1     CALL    TIM        ;Zeitverzoeg
        DJNZ    ST1
        JP      AUS

TIM     PUSH    BC         ;UP ZEITSCHLEIFE
        LD      C,0
TIM0    LD      B,0
TIM1    DJNZ    TIM1
        DEC     C
        JR      NZ,TIM0
        POP     BC
        RET
AZ      DEFB    0          ;Zwischenspeicher fuer Ausgabewert
TAB     DEFW    0204H      ;PIO-INIT, ATN LOW
        DEFW    6724H      ;Geraet 4=Listener, Sekundaeradr.
        DEFW    0FF03H     ;ATN HIGH , Ende-Kennung
```

T3 Der Sender ist bereit, das Byte auszugeben, CLOCK = High (T3-T2 ≦ unendlich).
T4 Ist der Empfänger bereit, wird die DATA-Leitung frei, das heißt DATA = High (T4-T3 ≦ unendlich).
T5 Das LSB des Bytes wird auf die Datenleitung gelegt, und CLOCK geht auf Low (T5-T4 = typisch 40 μs).
T6 Mit der LH-Flanke des Senders auf der CLOCK-Leitung übernimmt der Empfänger das Bit (T6-T5 = typisch 70 μs).
T7 Der Sender stellt das nächste Bit bereit und setzt CLOCK = Low (T7-T6 = typisch 20 μs).
T11 Der Sender setzt DATA = High und CLOCK = Low.
T12 Der Empfänger bestätigt die Übernahme des MSB durch DATA = Low (T12-T11 ≦ 1 ms).
T13 Der Sender ist bereit, das nächste Byte auszugeben (T13-T12 ≦ unendlich).
Die Bytequittierung unterscheidet sich für Daten- und Befehlsbytes nicht /2/.

Befehlsbytes
Es gibt die globalen Befehle 3FH (UNLISTEN) und 5FH (UNTALK), die aus einem Byte bestehen und sich an alle Peripheriegeräte wenden. Zweibytebefehle bestehen aus Primär- und Sekundäradresse.
Primäradresse
Das untere Halbbyte selektiert mittels der Gerätenummer das Gerät, während das obere die Funktion einstellt. (2XH, 3XH schalten auf LISTEN, 4XH, 5XH schalten auf TALK.)
Sekundäradresse
Sie setzt das Gerät in bestimmte Funktionsmodi, zum Beispiel werden beim Drucker Zeichensätze umgeschaltet. Die Sekundäradresse ist gleich oder größer 60H, die unteren 5 Bit kennzeichnen den Modus.

Software
Der Treiber wurde für die Ansteuerung eines Druckers SP180VC (Seikosha) entwickelt. Aufgrund der Besonderheit, daß eine Initialisierung des Druckers am Peripheriebus notwendig ist, wurden die Funktionen PIO (zur PIO-Programmierung, Byte 04), RESET (Drucker-Reset, 05), ATN = Low (02) und ATN = High (03) vorgesehen (PIO und RESET brauchen nicht über den Treiber erzeugt werden, sondern können auch systemgerecht installiert werden). Mit Hilfe des Programms **DRUCKERINITIALISIERUNG** wird die Bytefolge 04H 02H 24H 67H 03H abgearbeitet, die den Drucker am Bus aktiviert.
Weitere Programmteile des Treibers sind der Funktionsverteiler **MAIN**, **FRAM** für die Bytequittierung und **DOUT** zur seriellen Ausgabe eines Bytes.

Literatur

/1/ Albrecht, M.: DISKMON-AIM mit Floppy-Disk VC1541. 65xx MICRO-MAG
/2/ Löhr, R.: IEC: Die seriellen Busroutinen. 65xx MICRO-MAG

Dr. Ralf-Henning Seidenspinner

Bandlisten beim KC 87

Auch bei den Kleincomputern Z 9001, KC 85/1 und KC 87 kann der Kassetteninhalt mit einem Basic-Programm ermittelt werden, wie das folgende Listing zeigt. Von Basic-Programmen werden Namen und Schutzcode (in Form der ersten drei Bytes des FCB als Grafikzeichen) ausgegeben, von Maschinenprogrammen Anfangs-, End- und Startadresse sowie Name und Typ. Abgebrochen werden kann das Programm, wenn man während des Lesens eines Blocks die STOP-Taste gedrückt hält.
Hinweis: Es erfolgt keine Prüfung auf Lesefehler.

Reinhard Wobst

```
LIST

10 REM "tape list" R.Wobst 20.1.88
20 CALL *F593: POKE 107,0: N=0: RT$="   "+CHR$(13)+CHR$(11)
30 PRINT N;RT$;: DOKE 27,128: CALL *FF59: CALL *FAE3
40 N=PEEK(107): IF N>1 THEN30:ELSE IF N=0 THEN60
50 PRINT SPC(15);: FOR I=128 TO 138: GOSUB120: NEXT: PRINT: GOTO30
60 FOR I=145 TO 149 STEP 2: GOSUB80: PRINT " ";: NEXT
70 FOR I=128 TO 138: GOSUB120: GOSUB130: NEXT: PRINT: GOTO30
80 A=PEEK(I+1): GOSUB90: A=PEEK(I)
90 B=INT(A/16): A = A-16*B: GOSUB100: B=A
100 IF B>9 THEN B=B+7
110 PRINT CHR$(B+48);: RETURN
120 PRINT CHR$((32+PEEK(I)+ABS(PEEK(I)-32))/2);: RETURN
130 IF I=135 THEN PRINT ".";: RETURN: ELSE RETURN
OK
```

Programm Rollen rechts

Diese kurze MC-Routine für U 880 rollt den Inhalt des Bildschirms um genau ein Pixel nach rechts.
Dabei wird eine Bildschirmfläche von 255 × 255 Pixeln bewegt. Das Programm ist in verschiebbarem MC geschrieben und somit auf jedem freien Speicherbereich lauffähig. Die Speicherplatzbelegung dient nur als Vorschlag.

```
000    CD18 F0 FD E5 F5 21 00
000B   80 06 FF C5 E5 FD E1 FD
0010   7E 1F 0F 3F 3F 06 20 CB
0018   1E 23 10 FB C1 10 EC F1
0020   FD E1 CD1B F0 C9 00 00
```

Die Eingabe des Programms kann über Modify auf einen beliebigen Speicherbereich erfolgen. Eine andere Variante wäre das Einlesen von Basic aus (siehe Bild).
Es existieren folgende Optionen:
POKE < Startadresse dez. > + 19,X
X = 55 für herausrollen (Standard)
X = 63 für rundumrollen
X = 0 für invers rundumrollen

POKE < Startadresse dez. > + 8,Y
Y = Anfangszeile Rollfenster + 128 (Standard ganzer Bildschirm, nur gerade Zahlen < 30, 1 Rollzeile = 8 Pixel hoch)
POKE < Startadresse dez. > + 10,Z
Z = (Zeilenanzahl im Rollfenster) * B (Standard ganzer Bildschirm, nur gerade Zahlen < 30)

ACHTUNG:
Zeilenanzahl immer ≦ (32–Anfangszeile Rollfenster)
Der Aufruf des Programms erfolgt von Basic aus:
CALL < Startadresse dez. > bzw.
CALL * < Startadresse hex. >

Thorsten Noske

```
65210 RESTORE 65300:STARTADRESSE = 0 :WINDOW 0,31,0,39:CLS
65220 FOR K= 0 TO 37
65230 READ A:POKE K,A
65240 NEXT K
65300 DATA 205,24,240,253,229,245,33,0
65310 DATA 128,6,255,197,229,253,225,253
65320 DATA 126,31,15,63,63,6,32,203
65330 DATA 30,35,16,251,193,16,236,241,253,225,205,27,240,201,0
65340 FOR K=0 TO 25:PRINT AT(K,K);"ROLLEN RECHTS":NEXT K
65350 POKE STARTADRESSE + 19,63
65360 FOR K=0 TO 255:CALL 0 :NEXT K
65370 POKE STARTADRESSE + 10,16: POKE STARTADRESSE + 19,0
65380 FOR I=2 TO 28 STEP 2:POKE STARTADRESSE + 8,I+128
65390 FOR K=0 TO 255:CALL 0 :NEXT K
65400 NEXT I
```

Eine Bemerkung zu COPY Version 4.3

Auf einer Grundkassette für den Informatikunterricht in Schulen wird u. a. das Programm COPY Version 4.3 vertrieben, mit dem alle Files mit fortlaufenden Blocknummern kopiert werden können. Sind die Blocknummern nicht fortlaufend, wie z. B. bei OS-SAVE, so arbeitet COPY nicht oder fehlerhaft.
Um auch solche Files kopieren zu können, brauchen nur zwei Bytes verändert zu werden. Es sind dies:

Adresse	alt	neu
4B1	2A	C7
4E4	28	18

Beim so modifizierten Programm muß die erste Anfrage „Save option?“ stets mit ENTER beantwortet werden. Bei Lesefehlern springt COPY jetzt zum Warmstart des Systems, das heißt, das Lesen muß wiederholt werden.
Der Name des gelesenen Files steht ab 5C2H und kann mit einem entsprechenden Monitorprogramm vor dem Auslagern verändert werden.

Reinhard Wobst

Die intelligente Maschine

Der Computer als Experte

von M. Roth, Urania-Verlag Leipzig, Jena, Berlin, 1988, 96 S.; DDR 7,50 M

Die Mittel und Möglichkeiten der Künstlichen Intelligenz, eines Teilgebietes der Informatik mit starken Beziehungen zu anderen Wissenschaftsdisziplinen, entwickeln sich mit großer Geschwindigkeit, werden immer stärker theoretisch fundiert und zunehmend relevant für die Lösung von Problemen in (gemischten) Mensch-Maschine-Systemen. Dabei werden viele Probleme heiß diskutiert und von unterschiedlichen Standpunkten aus auch verschieden beantwortet – zum Teil stark emotions- und weniger kenntnisbeladen.

In diesem Problemfeld nimmt das vorgelegte Büchlein eine wertvolle und streitbare Position ein; wertvoll, weil es Laien auf diesem Gebiet ermöglicht wird, sich einen sachlich gut fundierten Überblick über die Aufgaben und Lösungen der Wissensverarbeitung und der Expertensysteme zu verschaffen. Streitbar, weil es akzentuiert die Möglichkeit und Sinnfälligkeit der Schaffung intelligenter computergestützter Systeme bejaht, begründet und verteidigt.

In geschickt komprimierter Darstellung werden sowohl die technischen Entwicklungen der Computerwelt als auch theoretische Grundlagen, vor allem aus der Logik, vorgestellt, die zusammen mit einigen philosophischen Hintergründen folgerichtig zum Begriff des Expertensystems führen. Nach Darstellung der Ziele und der Architektur derartiger Systeme wird an interessanten Beispielen dargestellt, welche Leistungen gegenwärtig erreichbar und welche künftig zu erwarten sind.

Die Darstellung läßt die ungeheure Spannweite dieser Problematik, von technischen und theoretischen Grundlagen bis zu moralischen, ethischen und philosophischen Fragestellungen, ahnen und weckt den Optimismus zu ihrer Bewältigung.

Eine interessante, zu Diskussionen anregende Lektüre, keine ausgereifte Darstellung für den Fachmann, vielmehr für einen breiten Leserkreis bestimmt, der am wissenschaftlich-technischen Fortschritt interessiert ist.

Prof. Dr. Ch. Posthoff

MS-DOS für Fortgeschrittene

von R. Duncan, Vieweg-Verlag Braunschweig, 1987, 473 S., ISBN 3-528-04539-6

Dieses Buch knüpft an die Tradition guter amerikanischer Fachbücher an. Es vermittelt anschaulich eine Fülle von Kenntnissen, deren Erwerb durch die Art der Darstellungsweise Freude beim Studium bereitet. Dabei bürgt schon der Name des Autors für Qualität. Ray Duncan ist durch eine ganze Reihe von Publikationen in diesem Genre bekannt. Er vermittelt mit diesem Buch in zwangloser Form und mit der Präzision des Könners Kenntnisse vom Umgang mit dem MS-DOS-Betriebssystem bei der systemnahen Anwendungsprogrammierung in Assembler und C. Mehrere Assembler-Programme illustrieren exemplarisch, wie die MS-DOS-Mechanismen funktionieren und wie sie praktisch gehandhabt werden. Zum Beispiel gibt es ein Terminal-Emulator-Programm, das anschaulich den Umgang mit der seriellen Schnittstelle zeigt. In abgewandelter Form können viele der mitgeteilten Programmkonstrukte zur Lösung ähnlich gearteter Probleme in der Praxis verwendet werden. Der im Kapitel „Installierbare Schnittstellentreiber“ vorgestellte Muster-Device-Driver, der leicht an eigene Bedürfnisse angepaßt werden kann, ist ebenfalls in diesem Zusammenhang erwähnenswert.

Bei „MS-DOS für Fortgeschrittene“ handelt es sich um die deutsche Übersetzung einer amerikanischen Ausgabe von Microsoft-Press, was dieses Buch auch als Referenzhandbuch autorisiert. Ob dieser oder ein anderer Grund dazu bewogen hat, den zweiten Teil des Buches in englischer Sprache zu belassen, sei dahingestellt und dürfte vom fortgeschrittenen Leser nicht als Mangel empfunden werden. Dieser zweite Teil enthält eine Auflistung aller Systemaufrufe bis einschließlich DOS-Version 3.1 in Form eines Referenz-Teils, der jeden DOS-Interrupt und seine Funktionen exakt beschreibt. Ebenso exakt sind die IBM-BIOS-Funktionen und der Lotus/Intel/Microsoft-Speichererweiterungs-Standard dargestellt. Der Referenzteil läßt an Vollständigkeit kaum etwas zu wünschen übrig; was ihn darüber hinaus auszeichnet, ist eine absolute Detailtreue und hohe Informationsdichte. Der fortgeschrittene Programmierer findet hier feinste Softwaredetails, untersetzt mit kurzen Beispielen, die, im Zusammenhang mit anspruchsvoller Hardwareliteratur zum IBM-PC betrachtet, ein rundes Bild der perfekten Handhabung eines PCs geben können. Damit ist dann auch der Leserkreis für das Buch charakterisiert, der sich wohl vornehmlich in den Reihen der Systemprogrammierer und engagierten PC-Anwender finden läßt, die es ganz genau wissen wollen. Wer ständig nach dem Wie fragt und den Sachen stets auf den Grund geht, der kommt bald an den Punkt, an dem ihm möglicherweise dieses Buch zu einem exklusiveren Verständnis der internen Vorgänge in seinem PC verhelfen kann. *Th. B.*

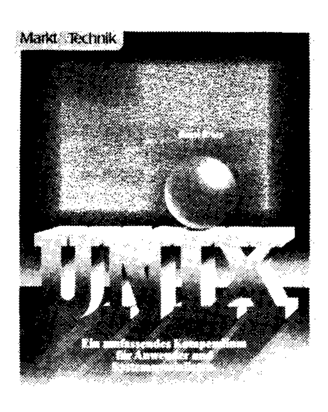
UNIX

Ein umfassendes Kompendium für Anwender und Systemspezialisten

von H. Drees, Verlag Markt & Technik, München 1988, 570 S., 89,– DM, ISBN 3-89090-494-7

Einhergehend mit der stark wachsenden Akzeptanz und Bedeutung des Betriebssystemkonzeptes UNIX erscheint international eine Flut von Büchern zu dieser Thematik, die durch sich gegenseitig übertreffende Ansprüche um Käufer werben.

Nach den Worten des Autors wurde der vorliegende Band für alle, die UNIX kennenlernen möchten, die



täglich mit UNIX umgehen, und auch für jene, die UNIX zwar schon in groben Zügen kennen, aber noch mehr über dieses mächtige Betriebssystem erfahren möchten, geschrieben.

Das Buch enthält demzufolge eine Einführung in das UNIX-Konzept (ca. 40 Seiten), die Behandlung des UNIX-Dateisystems (ca. 30 Seiten), eine Beschreibung des UNIX-Prozeßsystems (ca. 20 Seiten), die Behandlung der UNIX-Shell-Kommandointerpreter (ca. 20 Seiten), eine Einführung in die Programmiersprache C (ca. 50 Seiten), eine Kurzübersicht über die Kommandos der Editoren ed, ex, vi (ca. 25 Seiten), eine Einführung in die Dokumentationsaufbereitung mit „nroff“ und „troff“ (ca. 30 Seiten) und eine sehr ausführliche Behandlung des Standard-Kommandosatzes von UNIX-System V (ca. 300 Seiten).

Wie aus dieser Inhaltsübersicht hervorgeht, liegt der Schwerpunkt des Buches beim Standard-Kommandosatz. Beschrieben werden alle gebräuchlichen Kommandos, einschließlich ihrer Optionen, in alphabetischer Reihenfolge, so daß wichtige Informationen für den Leser schnell zugriffsbereit vorliegen. Hervorzuheben ist die sorgfältige typografische Gestaltung des Bandes, die auch wesentlich für seine guten Gebrauchswerteigenschaften mitverantwortlich ist.

Dem Rezensenten erscheint das Buch, trotz seines anderslautenden Untertitels, weniger für UNIX-Systemspezialisten geeignet (dafür ist an vielen Stellen einfach zu wenig Platz zur Behandlung von diesen Leserkreis betreffenden Themen), sondern vielmehr ein wirklich umfassendes Kompendium für den UNIX-Anwender zu sein. Er findet hier in einem handlichen Band zusammengestellt viele wichtige Informationen, die dieses Buch als ständiges Arbeitsmittel am UNIX-Terminal geeignet erscheinen lassen.

Dr. L. Claßen

FORTH

Einführung und vollständiger Programmierkurs in Forth,

von P. Kail, R. Oldenbourg Verlag München Wien 1988, 193 S.

Der Autor stellt sich die Aufgabe, dem Leser den Zugang zu dem hochgradig interaktiven und effektiven Programmierwerkzeug FORTH zu ermöglichen. Sein Buch wendet sich an FORTH-Laien und „... setzt keinerlei Hardwarekenntnisse voraus.“ In der angegebenen Reihenfolge werden folgende Komplexe behandelt: Die Grundstrukturen von FORTH; der Stapel; die Definition neuer Wörter; Variablen: IF, ELSE, THEN; Schleifenstrukturen; Doppelt formatierte Arithmetik; Wortdefinitionen; Massenspeicherung; Strings; das Für und Wider von FORTH.

Im Anhang befindet sich eine Liste wichtiger FORTH-Wörter sowie eine Liste der ASCII-Codes. Den Schluß des Buches bildet ein umfangreiches Sachwortverzeichnis.

Es wird der Versuch unternommen, ohne tieferes Eingehen auf die Hardware dem Leser FORTH möglichst umfassend nahezubringen. Daraus resultiert, daß die hardwareabhängigen Teile, wie das Erzeugen von Primaries, z. B. mittels In-Line-Assembler, nicht abgehandelt werden. Das stellt aber, entsprechend dem Anliegen des Buches, keinen Nachteil dar.

Eingestreute kurze BASIC-Programme sollen den Zugang zu FORTH erleichtern und die Besonderheiten der Sprache demonstrieren. Die FORTH-Programmbeispiele wirken oft recht konstruiert. Dessen scheint sich der Autor auch bewußt zu sein: „Obwohl dieses Beispiel etwas albern ist ...“ (S. 52).

Bei einigen Beispielen wurden Trennzeichen zwischen Wörtern vergessen; wer die Beispiele nachvollziehen möchte und in seinen Rechner eintippt, wird sicher von seinem FORTH-System belehrt werden.

Der Autor sieht bei der Definition mittels CONSTANT den Vorteil, daß die „Zahl in einem Vorgang abgerufen werden kann“ (S. 141) und den Nachteil, „daß der Wert der Zahl nicht so einfach geändert werden kann“ (S. 141) bzw. spricht von „der Schwierigkeit, ihn zu ändern“ (S. 50).

Wir sehen einerseits keine Schwierigkeit, andererseits aber auch keine Notwendigkeit, den Wert einer einmal definierten Konstante zu ändern.

Dem Leser wird dann auf Seite 161 erklärt: „Die meisten Programmierer sind an die post-fix-Notation und nicht an die umgekehrt polnische Notation gewöhnt“. Wenn „post-fix“ durch „infix“ ersetzt wird, ist diese Aussage sicher zu akzeptieren. Die Übertragung aus dem Englischen weicht in einigen Fällen von den geläufigen FORTH-Termini ab, z. B. Definierworte statt definierende Worte oder doppelt formatierte Arithmetik statt doppeltgenauer Arithmetik.

Die Rezensenten sind der Meinung, daß das vorliegende Buch, trotz einiger Schwachstellen, einen interessanten Versuch darstellt, die FORTH-Programmierung dem Einsteiger zu erschließen.

Dr. H. Finsterbusch/Dr. W. Horn

Simulator für Einchipmikrorechner

Als Alternative zum Emulator, einer vergleichsweise aufwendigen Hardwarelösung, wurde im Computer-Club Karl-Marx-Stadt eine Softwarelösung, der Simulator, entwickelt. Mit Hilfe des Simulators ist es möglich, die EMR-Software auf einem CP/M-Rechner X (ZX SPECTRUM) zu testen. Das auf dem CP/M-Rechner editierte und mit einem entsprechenden Assembler übersetzte EMR-Programm kann dabei auch schrittweise abgearbeitet werden, indem alle wesentlichen Funktionen des EMRs nachgebildet werden. Alle Register des EMRs werden ständig mit aktuellem Inhalt angezeigt. Der Inhalt der Speicherzelle, auf die der Programmzähler zeigt, wird als Befehl interpretiert, rückübersetzt und dargestellt.
Bedienerkommandos existieren für:
Rückübersetzen,
Listen des Speicherinhaltes,
Ändern des Programmzählers,
Ändern des Inhaltes des Speichers,
Überschreiten des Inhaltes der Register
Breakpointadresse setzen,
Abarbeiten des aktuellen Befehls,
Programmlauf bis entweder der Breakpoint erreicht ist
oder die Space-Taste betätigt wird,
Programmzähler erhöhen und verringern,
Interruptsimulation,
Hardcopy.
Mit Hilfe des Simulators ist es somit möglich, Programme für EMR ohne Zusatzhardware auf einem beliebigen CP/M-fähigen Rechner (z. B. PC 1715 oder BC 5120) oder auf dem ZX-SPECTRUM auf logische Richtigkeit zu testen. Der Simulator erfüllt softwaremäßig annähernd alle Funktionen, die beim Emulator hardwaremäßig erreicht werden. Die Ausnahme bildet der Echtzeitlauf.

Computer-Club Karl-Marx-Stadt, Kennwort „Z8-Simulator", PSF 607, Karl-Marx-Stadt, 9010

Pestel

Softwareprojekte für Verwaltungsarbeit

Zur Rationalisierung der Verwaltungsarbeiten in der LPG Milchproduktion Prießnitz wurden folgende Softwareprojekte erarbeitet:
– Lagerbestandsführung mit den Hauptfunktionen Materialstammdaten, Lieferübersicht, Materialbestellungen, -reservierungen, -abgänge und -zugänge sowie mit umfangreichen Informationsrecherchen
– mobile Technik mit den Hauptfunktionen Stammdaten, Abrechnung, Kraftstoffverbrauch, Reparaturdaten und Reparaturpläne sowie mit umfangreichen Informationsrecherchen
– Personalstatistik mit den Hauptfunktionen Personalstammdaten, Berufe, Befähigungsnachweise, Organistionen, Auszeichnungen, Urlaubsstatistik, Krankheitsstatistik und Freistellungen sowie mit umfangreichen Informationsrecherchen
– Grundmittelstatistik
– Terminkontrolle
– Tierbestandsübersicht (Rinder) einschließlich Milchleistung und Zuchthygiene.
Diese Softwareprojekte werden für die Betriebssysteme SCP, DCP und MS-DOS zur Nachnutzung angeboten. Da die Projekte auf BC A 5120; PC 1715, A 7100, A 7150, EC 1834, PC 1512 und PC 1640 lauffähig sind, können auch Betriebe, die unterschiedliche Hardware verwenden, einheitliche Projekte anwenden.

LPG Milchproduktion Prießnitz, Prießnitz, 4801; Tel. Janisroda 285

Seidel

Lohn-/Gehaltsprojekt und Themenabrechnung für F/E

Für alle Nutzer des **Lohn- und Gehaltsprojekts (auf BC 5120/30)** vom VEB Robotron-Vertrieb Erfurt bieten wir folgende Änderungen und Programmergänzungen an:
– Druck der Gehalts- und Lohnnachweise mit Erläuterungstexten auf Normalpapier
– Durchführung der Lohn- und Gehaltsrechnung zu unterschiedlichen Terminen mit einer Stammdatei
– Sicherung der exakten Monatsabrechnung für alle Beschäftigten sowie die Einordnung der Zu- und Abgänge in das Lohnkonto durch die zusätzlichen Stammdatenfelder *Vorauszahlung Vormonat, Vorauszahlung laufender Monat*
– umfassendes Korrekturprogramm für die monatlichen Ergebnisdaten der Stammdatei
– zusätzliche Auswertungslisten: Auszahlungsliste mit Geldstückelung, FDGB- und Solidaritätsbeiträge einschließlich Markenbestellung und -abrechnung, Stammdatenauswertungen.
Erfahrungswerte und Informationen zum kompletten Programmpaket, wie Personalstammdaten, monatliche Abrechnungsdaten, Abrechnungsalgorithmus, Krankengeldrechnung, Krankenstatistik, Auswertungslisten und kumulatives Lohnkonto, können über unsere Abteilung TA eingeholt werden.
Das Programm **F/E-Themenabrechnung** ist ein menügesteuertes, modular aufgebautes System für den PC 1715 bzw. A 5120/A 5130 (REDABAS) mit den Komponenten
– Aufbau und Pflege der Personalstammdatei
– Aufbau und Aktualisierung der Themenstammdaten
– Themenabrechnung
– Umbuchung von Leistungen
– Drucklisten
– Sicherung der Datenbestände.
Ausgehend von der Personal- und Themenstammdatei werden die monatlichen Zeitnachweise der F/E-Mitarbeiter erfaßt. Dabei erfolgt eine interne Prüfung der Gültigkeit der Themen- und Personalnummer und die Prüfung auf bereits abgeschlossene Themen. In die Abrechnung einbezogen werden die Lohnkosten (resultierend aus dem Zeitnachweis), die themenbezogenen Gemeinkosten, die Fremdleistung, die Materialkosten und die themengebundenen Grundmittel. Die Auswertung (analog dazu die entsprechenden Drucklisten) gliedert sich in: monatliche Themenabrechnung nach Themennummer und Stundenaufwand je Mitarbeiter, Gesamtsumme aller Themen pro Hauptabteilung, Stand der Themen im Planjahr und Stand der Themen ab Themenbeginn.

VEB Institut für Spielzeug Sonneberg, PSF 155, Sonneberg, 6400; Tel. 4 35 12

Zinner

IFSS-Druckerweiche

Die Mehrzahl von Druckern mit IFSS-Schnittstelle ist fest an einen Rechner angeschlossen, wird nur einen Teil der Rechnerzeit benötigt und kann nur durch Umstecken des Anschlußkabels an einen zweiten Rechner benutzt werden.
Durch die Druckerweiche wird der Betrieb von einem Drucker an drei Rechnern ermöglicht. Das Drucken kann ohne Umstecken der Kabel zeitversetzt erfolgen. Der Anschluß der Geräte ist unabhängig von Rechner- und Druckertyp und wird nur durch die Übereinstimmung der Schnittstellendaten von Rechner und Drucker festgelegt. Die Platine kann auch mit zwei Rechnern betrieben werden. Das Prinzip der Schaltung ist jedoch auch für eine größere Anzahl von Rechnern anwendbar. Die Stromversorgung der Weiche kann aus den angeschlossenen Druckern oder aus einem separaten Netzteil erfolgen.

Handelshochschule Leipzig, ORZ, Bereich Elektroniklabor, Markgrafenstraße 2, Leipzig, 7010; Tel. 74 81

Hoffmann

Symbole und Variantenkonstruktionen für PCCAD

Für die Anwendung auf 16-Bit-PCs wurde unter der Grafiksoftware PCCAD folgende Zusatzsoftware in Form von Symbolen (Form A und B) und Variantenprogrammen entwikkelt:
1. Symbole für Hydraulikpläne
– Symbole für Funktionsschaltpläne nach TGL 8672
– Symbole für Bauschaltpläne (Plattenverschraubungen) nach TGL 26 215/50
2. Variantenkonstruktionen für allgemeinen Maschinenbau
– Schraubverbindungen nach TGL 0-912, 0-931, 0-84, 5683
– Sicherungsringe nach TGL 0-471, 0-472
– Hydraulische Arbeitszeitzylinder der Form B1, B2, C1, S1, S2 nach TGL 10 906
– Kolbenstangenköpfe der Form A und B nach TGL 21 549
– Rohrverschraubung nach TGL 8277, 0-2353
– Rohrschellen nach TGL 39 247.

VEB Plasttechnik Greiz, HA Entwicklung und Konstruktion, Plauensche Straße 40–42, Greiz, 6600; Tel. 7 93 08

Großwig

Zeichnen im alphanumerischen Dialog

Das Programmpaket **CADki** ermöglicht die Zeichnungserstellung für Einzelteile des Maschinenbaus nach der Methode der Variantenkonstruktion (s. a. Maschinenbautechnik 12/87). Vertrieben werden zur Zeit die Programmteile
– Rippe
– Buchse, Scheibe
– Zeichnungsbeschriftung (Verzahnungstabellen u. ä.).
Hardwarevoraussetzung: 8-Bit-Computer BC 5120/30, PC 1715, PRG 700 oder 16-Bit-Computer AC 7100/50, EC 1834 sowie der Plotter SPL 430.
Die Ausgabe für Plotter K 6411 wird vorbereitet. Der Plotter K 6418 bzw. ein Grafikdrucker sind nicht geeignet. Die Programme arbeiten unter den Betriebssystemen SCP oder DCP. Der Vertrieb der Programmteile
– Welle
– Platte
– Profilteile
– Rotateil mit Verzahnung
erfolgt ab 1989.

VEB Schraubenwerk Karl-Marx-Stadt, BT Ratiomittelbau, Abteilung TRTKS3, Südring 1, Karl-Marx-Stadt, 9030; Tel. 80 32 21

Kirchhübel

TP für elektronische Schreibmaschinen

Zum Herstellen von repräsentativen Schriftstücken bietet sich der Anschluß einer elektronischen Schreibmaschine als Schönschriftdrucker an.
Für den Einsatz einer elektronischen Schreibmaschine (nachgerüstet mit der vom Zentralinstitut für Kernforschung Rossendorf angebotenen Interfacesteckeinheit; siehe MP 4/88) als Drucker im Textprogramm TP (unter CP/A oder SCP) wurden verschiedene Versionen des Textprogramms (TPS10, TPS20) installiert. Mit diesem Programm werden alle Druckfunktionen des Textprogramms wie Halbzeile hoch/tief, Doppeldruck, Durchstreichen, Unterstreichen, Rückschritt, Wagenrücklauf ohne Zeilenschaltung, Normalschrift (1/10 Zoll), alternativer Zeichenabstand (1/12 Zoll) unterstützt. Weiterhin ist die Funktion Schattenschrift möglich. Die beim Textprogramm möglichen vier Anwendersteuerzeichen werden bei der Version TPS10 ignoriert. Version TPS20 nutzt alle verfügbaren Anwendersteuerzeichen, so daß noch zusätzlich Möglichkeiten zur Druckgestaltung (Proportionalschrift, Zeichenbreite 1/15 Zoll, Zeilenabstand 1/6 Zoll, Zeilenabstand 1/4 Zoll) zur Verfügung stehen.
Eine ausführliche Dokumentation mit Demonstrationstext ist vorhanden.

Technische Hochschule „Carl Schorlemmer" Leuna-Merseburg, Direktorat für Forschung, Otto-Nuschke-Straße, Merseburg, 4200; Tel. 4 60

Dr. Domaratius

UNIX-Maskengenerator/-prozessor

Das entwickelte Programmsystem ermöglicht es, in einfacher Weise Bildschirmmasken für UNIX-kompatible Betriebssysteme zu erzeugen und einzusetzen. Eine Maske wird in einer einfachen Sprache beschrieben und in Form eines Maskenbeschreibungsfiles als Textfile gespeichert. Dieses File kann mit Hilfe eines Texteditors oder interaktiv mit dem Maskeneditor erzeugt bzw. modifiziert werden. Durch den Maskengenerator wird dieses File interpretiert und die Maske auf dem Bildschirm generiert. Der Maskenprozessor unterstützt alle Dateneingaben bzw. -ausgaben, startet Datenprüfroutinen, steuert die Kursorpositionierung und erzeugt Ausgabefiles aus den eingegebenen Daten. Die Ausgabefiles bilden die Schnittstelle zu anderen vom Nutzer zu schreibenden Programmen (z. B. für Datenbankzugriffe). Durch verschiedene Datenprüfroutinen kann vom Maskenprozessor die Richtigkeit der Eingabedaten festgestellt werden. Die Verwendung von Prüfroutinen und Steuerregeln ermöglichen es auch, in Abhängigkeit von den eingegebenen Daten, Eingabefelder in unterschiedlicher Reihenfolge zu durchlaufen. Für das Erfassen von Listen können mehrere Eingabefelder zu Scroll-Bereichen zusammengefaßt werden. Aus mehreren Masken kann ein Maskennetz gebildet werden. Dieses wird wieder durch ein Maskennetzbeschreibungsfile beschrieben. Der Maskennetzprozessor steuert die Abarbeitungsreihenfolge der einzelnen Masken sowie weiteren vom Nutzer angegebenen Programmen.

VEB Nachrichtenelektronik Leipzig „Albert Norden", Abteilung EK5, Melscherstraße 7, Leipzig, 7027; Tel. 6 83 33 82

Müller

Diskettenpflege für den PC 1715

Es werden die von POWER und ähnlichen Dienstprogrammen bekannten Operationen zur Diskettenarbeit angeboten. Neu ist die völlig anders gestaltete Art der Dialogführung. Der Benutzer kann sich seine gewünschte Datei aus dem am Bildschirm angezeigten Verzeichnis durch Anwahl mittels Kursertasten auswählen und durch Drücken einer Taste eine gewünschte Operation auslösen. Operationen mit mehreren Dateien sind ebenfalls möglich. Diese können per Hand oder mittels einer Dateimaske markiert werden. Durch eingebaute Fehlerprüfung gibt es keine unkontrollierten Reaktionen (z. B. beim Befehl TYPE bei POWER). Neu ist, daß man in einer beliebigen ASCII-Datei vorwärts und rückwärts blättern kann.
Ebenfalls eingebaut sind ein Unterprogramm zur Wiederherstellung gelöschter Dateien (der Befehl RECLAIM arbeitet bei POWER übrigens fehlerhaft) sowie ein komfortabler Editor zum problemlosen Editieren beliebiger Dateien. Das Programm wurde speziell für den PC 1715 unter dem Betriebssystem SCP entwickelt.

VEB Chemiekombinat Bitterfeld, Abt. FI/G, Zörbiger Straße, Bitterfeld, 440; Tel. 7 62 50

Dr. Grahn

Serielle Kopplungskarte

An der Sektion TdE/E der TH Wismar wurde für die serielle Kopplung beliebiger Rechner mit K 1520-Bussystem eine Leiterkarte mit der Bezeichnung **SIOCPC** entwickelt. Die Karte realisiert sowohl eine Standard-IFSS-Schnittstelle als auch eine abgerüstete V.24-Schnittstelle (nur Datensende- und -empfangsleitung TxD, RxD). Die Sender- und Empfängerstromschleifen können aktiv oder passiv betrieben werden.
Technische Eigenschaften:
– Format 170 mm × 95 mm
– Betriebsspannung +5 V und +12 V
– DC/DC-Wandler für –12 V
– Adreßdekodierung für 8 Bit, umschaltbar auf 16 Bit
– Bei Verwendung von 4-MHz-Bauelementen und weiterer an der THW entwickelter Hardware (Anpassung Expansionsbus an K 1520-Rechnerbus, Stromversorgungsmodul, Anschluß eines Diskettenlaufwerkes 5 1/4 Zoll) ist die Nutzung als Ergänzungsbaugruppe für den *Schneider CPC* über den Expansionsport möglich.
Bei Verwendung eines CP/M-kompatiblen Betriebssystems kann ein im Lieferumfang enthaltenes Kommunikationsprogramm genutzt werden.
Es besteht die Möglichkeit, 8- und 16-Bit-Mikrorechner über Standardsoftware zu koppeln.
Für diese Baugruppe stehen eine nachnutzbare zweiseitige, nicht durchkontaktierte, unbestückte Leiterplatte, eine Dokumentation sowie ein Kommunikationsprogramm zur Verfügung.

Technische Hochschule Wismar, Sektion TdE/E, Wissenschaftsbereich Grundlagen der Elektrotechnik/Elektronik, Phillip-Müller-Straße, Wismar, 2400; Tel. 5 33 33

Mecker

Speichern von Bildschirmseiten

Zum Abspeichern von Bildschirmseiten auf dem A 7150 und deren Aufruf wurde ein speicherresidentes Programm unter DCP entwickelt.
Das als COM-Datei ausgeführte Programm ist durch einen umgelenkten Tastaturinterrupt aufrufbar. Das in Assembler (MASM 1.0) geschriebene Programm unterstützt alle CGA-Bildschirmmodi. Es läßt sich aus anderen Programmen heraus starten und kehrt nach dem Abspeichern des Bildschirminhaltes an den Ausgangspunkt zurück. Der Test des Programms erfolgte aus Turbo-Pascal 4.0, MS-Chart, Printmaster, REDABAS-3 und Basic heraus. Die zu jedem beliebigen Zeitpunkt erzeugbaren Bilddateien lassen sich in eigenen Programmen weiterverarbeiten. Installiert wird das Programm durch Aufruf oder Einbinden in die AUTOEXEC.BAT.

Amt für Standardisierung und Meßwesen, Abt. 0220, Fürstenwalder Damm 388, Berlin, 1162; Tel. 6 44 12 88

Sand

Programmiersprache FLOKC

Auf der Grundlage des Forth-Moduls des KC 85/3 wurde mit der Entwicklung zweier Versionen (4 KByte, ca. 70 Befehle und 8 KByte, ca. 100 Befehle) der listenverarbeitenden rekursiven Programmiersprache FLOKC (FORTH-LOGO-KC) der Versuch unternommen, insbesondere für die Informationsausbildung eine Programmiersprache anzubieten, die in ihrem Erscheinungsbild und ihren Programmeigenschaften unmittelbar an die Programmiersprache LOGO anschließt. Ihre Vorzüge:

- eine umfassendere Nutzung des Forth-Moduls des KC 85/2,3
- für Ausbildungszwecke wird eine Sprachstruktur verfügbar, die der rekursiven Programmierung angepaßt ist.
- Erprobung einer leicht verständlichen listenverarbeitenden Sprache, die konsequent zu einem Bottom-up-Programmierstil zwingt und strukturiertes Programmieren unterstützt
- Nutzung der Möglichkeiten der Turtle-Grafik, die u. a. eine Heranführung von Kindern an den Computer und das Programmieren fördern
- Test der Leistungsfähigkeit einfacher Programme zur symbolischen Formelmanipulation (z. B. symbolisches Differenzieren elementarer Funktionen)
- Verfügbarkeit über eine lisp-verwandte Sprachstruktur, die der Demonstration von Problemstellungen der KI dienen kann.

Pädagogische Hochschule „Karl-Liebknecht", Sektion Marxismus-Leninismus, Potsdam, 1570

Dr. Petsche

Verwalten von Diskettenbeständen

Programmfunktionen:
– Automatisches Archivieren und Rearchivieren
– Anzeigen, Drucken und Ändern des Archivs,
– Disketten mit Namen versehen
– Recherche nach Archivmerkmalen
– Archivieren leerer Disketten
– Anzeigen und Drucken des Diskettenstatus
– Help-Funktionen.
Unterstützung folgender Diskettenformate:
9 × 512 × 40 = 360 KByte
9 × 512 × 80 = 720 KByte
5 × 1024 × 80 = 780 KByte
sowie von Disketten mit 1,2 KByte.
Das Programm läuft unter DCP-kompatiblen Betriebssystemen und liegt in einer Turbo-Pascal- und in einer REDABAS-3-Variante vor.
Hardware-Voraussetzungen:
– 512 KByte RAM
– 1 Harddisk ≧ 20 KByte und mindestens 1 Floppy-Laufwerk K 5601 (Typ 1.6)
– A4-Drucker.

VEB Automobilwerk Eisenach, TVQ, Rennbahn 8, Eisenach, 5900

Haupt/Dr. Remmler

Macroassembler für Kleincomputer

Das Programm ist ein Assemblersystem für Kleincomputer der Typen KC 85/2/3, KC 85/1, KC 87, Z 1013 und ZX Spectrum. Es besteht aus dem Editor, dem Macroassembler und einfachen Testhilfen.
Der Editor ist bildschirmorientiert und wurde nach dem Vorbild von Wordstar gestaltet. Es werden die gleichen Kommandos verwendet. Auch die Blockkommandos Löschen (^KY), Kopieren (^KC), Verschieben (^KV), das Laden eines Blocks von der Kassette und das Einfügen an beliebiger Stelle (^KR), das Abspeichern eines Blocks auf der Kassette wie auch die Kommandos zum Suchen (^QF) sowie Suchen und Ersetzen (^QA) sind verfügbar. Ein zusätzliches Kommando (^J) erwartet die Eingabe eines Labels und sucht die Stelle im Text, wo das Label definiert ist. Auf diesem Wege lassen sich schnell Unterprogramme finden. Der Quelltext wird in komprimierter Form abgespeichert, um den knappen Speicherplatz besser ausnutzen zu können. Dadurch lassen sich bei etwa 23 KByte freiem Speicherplatz aus einem Quelltext etwa 4,5 KByte Maschinenprogramm erzeugen.
Der Assembler bietet die Möglichkeit der Definition und der Nutzung von Macros auch mit formalen Parametern. Mit den Pseudobefehlen IF, ELSE und ENIF können Textabschnitte eingeschlossen werden, die nur unter bestimmten Bedingungen mit assembliert werden sollen. Die beim Assemblieren gebildete Markentabelle kann auch beim Testen verwendet werden. Sie wird auch durch erneutes Editieren des Textes nicht zerstört. Die Testfunktionen ermöglichen den Aufruf von Programmen mit Initialisierung der Register und mit Anzeige der Registerinhalte bei Rückkehr sowie die Anzeige der Werte von Labels und Zahlenumrechnungen.

Karl-Marx-Universität, Institut für Biophysik, Liebigstraße 27, Leipzig, 7010

Rödenbeck

Wir suchen ...

... eine Lösung zur Kopplung des Kleinplotters XY 4140 (ČSSR) mit dem KC 85/3.

Pädagogische Hochschule „Dr. Theodor Neubauer", PSF 307 und 848, Erfurt, 5010; Tel. 53 62 96

Dr. Barth

... zur Nachnutzung Cross-Software für den AC 7150 (Betriebssystem SCP oder DCP), mit der das Entwickeln und Testen von Software für U 880 und U 882 möglich ist, sowie einen Emulator für U 882 zum Anschluß an den AC 7150 (einschl. Software).

VEB Elektromat Dresden, HA W4, Karl-Marx-Straße, Dresden 8080; Tel. 5 93 31 86

Lehmann

... Algol 60-Compiler für 16-Bit- oder 32-Bit-Mikrorechner.

VEB BMK Ost, Betrieb FPT, BT Berlin, Storkower Straße 134, Berlin, 1055; Tel. 4 34 23 04

Berger

... eine Lösung zum Anschluß eines Plotters vom Typ K 6418 an einen Schneider-PC unter MS-DOS 3.2.

VEB Organisations- und Abrechnungszentrum Berlin, Rechenstation, Boxhagener Straße 18, Berlin, 1034; Tel. 5 80 73 07/22

Hoth

Intel 80486 angekündigt

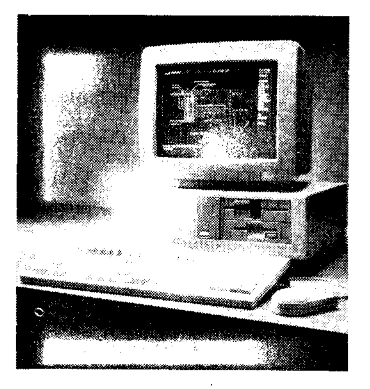
Wenige Wochen nach der überraschenden Vorstellung ihres ersten RISC-Mikroprozessors 80860 – über den wir demnächst ausführlicher berichten werden – und etwa zeitgleich mit dem Lieferstart des 80286-Prozessors mit 33 MHz Taktfrequenz kündigte die Firma Intel im April offiziell den neuen 32-Bit-Mikroprozessor 80486 an. Seine Leistung soll bei gleicher Taktfrequenz das 2- bis 3fache des Vorgängers 80386 betragen; bei 33 MHz also etwa 20 MIPS. Zu dieser Leistungsfähigkeit tragen ein Fünf-Stufen-Pipelining und 128-Bit-Datenpfade ebenso bei wie die Tatsache, daß der größte Teil der Befehle wie etwa bei RISC-Prozessoren mit nur einem Taktzyklus abarbeitbar sein soll. Wie bereits in unserer Ausgabe 12/88, S. 384, ausführlich beschrieben, hat der 80486 keine generell neue oder RISC-Architektur. Er integriert neben der Speicherverwaltungseinheit (MMU) nun auch die Gleitkommaeinheit (FPU) sowie einen internen Cache. Für Multiprozessorsysteme ist ein Controller in Vorbereitung, der Cachespeicher bis zu 512 KByte unterstützen kann. Während die 25-MHz-Version des 80486 Ende dieses Jahres in Stückzahlen lieferbar sein soll, wird die 33-MHz-Version erst Anfang 1990 erwartet.
Eine Ausführung des Chips in ECL-Technik, die nochmals eine Leistungssteigerung bis zum Faktor fünf ergeben soll, ist in Zusammenarbeit mit der Firma Prime für 1992 angekündigt. *MP*

Motorola 68030 mit 50 MHz

Wie bei Intel, so arbeitet man auch bei der Firma Motorola intensiv an der Fertigstellung eines neuen 32-Bit-Prozessors, des 68040. Er soll ebenso wie der 80486 etwa 1,2 Millionen Transistoren enthalten und die FPU, MMU sowie Caches integrieren. Auch legt man Wert auf Multiprozessorfähigkeit, die mittels Snoop-Controller gewährleistet wird. Als Leistung werden bei der 33-MHz-Version 24 MIPS angegeben. Die Auslieferung des Prozessors soll noch in diesem Jahr erfolgen. Gleichzeitig wird jedoch offensichtlich noch die Weiterentwicklung des 68030 vorangetrieben. So wurde jetzt, nur kurze Zeit nach der Ankündigung des 33-MHz-68030 (s. MP 1/89, S. 30), die 50-MHz-Version angekündigt. Der Prozessor ist in 1-μm-HCMOS-Technologie gefertigt und soll mit etwa 12 MIPS doppelt so schnell wie die 25-MHz-Variante und wesentlich schneller als der 33-MHz-80386 von Intel (7,5 MIPS) sein. *MP*

HP-Workstation mit 68040 geplant

Die Firma Hewlett Packard gehört zu dem kleinen Kreis von Computerproduzenten, die auf eine 50jährige Geschichte zurückblicken können. Im April 1939 wurde das Unternehmen von den beiden jungen Ingenieuren William R. Hewlett und David Pak-



kard als „Garagenfirma" gegründet und produzierte Oszilloskope und Meßinstrumente. Von vielen wird HP heute als Marktführer in der Meßtechnik angesehen. Der erste Computer kam 1966 zur Steuerung programmierbarer Meßgeräte, gefolgt 1968 vom ersten programmierbaren Tischrechner. Heute sind auch leistungsstarke PCs – das Bild zeigt beispielsweise den neuen Vectra QS/20, der eine Zwei-Bus-Architektur aufweist – und Workstations Schwerpunkte bei HP. Schon 1986 führte die Firma mit der sogenannten Precision Architecture das RISC-Prinzip in ihre Workstations ein.
Nachdem Ende 1988 bereits die Entwicklung der Workstation-Serie 9000 mit dem 33-MHz-Prozessor 68030 angekündigt wurde, will HP nunmehr der erste Hersteller sein, der den modernsten Mikroprozessor von Motorola, den 68040, einsetzt. Die Implementierungsarbeiten haben bereits begonnen.
Die kommenden Workstations sind für den Betrieb unter HP-UX – das der AT & T UNIX System V Interface Definition Issue 2 entspricht – optimiert und kodekompatibel mit den 68020- und 68030-gestützten Workstations der Familie HP 9000 Serie 300. Damit werden sie eine Leistungssteigerung ohne Modifikation bestehender Applikationssoftware erlauben. *MP*

Neuer 16-Bit-PC von Compaq

Während von der Compaq Computer Corporation wegen der guten Verbindungen zu Intel die Erstvorstellung eines PCs mit dem 33-MHz-Mikroprozessor Intel 80386 im Frühjahr erwartet wurde, präsentierte die Firma im April zunächst einen weiteren PC mit dem 16-Bit-Prozessor 80286. Compaq-Vice President Mike Swatey (Markt und Vertrieb) dazu: „Wir rechnen bis weit in die 90er Jahre mit einem beträchtlichen Marktanteil für die 286-Systeme, da diese Kategorie kleiner Desktop-PCs die Produkte auf Basis des 8088/8086-Prozessors im Bereich der professionellen Anwendungen ablösen wird." Der Deskpro 286e ist als Einstiegsmodell für professionelle Anwender besonders in Büroumgebungen gedacht. Der 80286-Prozessor taktet mit 12 MHz und kann optional durch einen 12-MHz-80C287- oder 8-MHz-80287-Koprozessor unterstützt werden. Der 1-MByte-RAM läßt sich unter Nutzung eines Steckplatzes bis auf 13 MByte erweitern.
Da sowohl der Standard-RAM als auch der Grafik-Controller, die Standardschnittstellen und der Disketten-Controller auf der Systemplatine enthalten sind, können insgesamt fünf freie Leiterkartensteckplätze geboten werden. In der Standardausstattung gibt es ein 5,25-Zoll-Floppylaufwerk mit 1,2 MByte und eine Festplatte mit bis zu 40 MByte (< 29 ms Zugriffszeit), als Option 3,5-Zoll-Floppylaufwerke mit 1,44 MByte, 110-MByte-Festplatten (< 25 ms) und 135-MByte Streamer.
Ebenfalls am 10. April stellte Compaq für seine Computer die Betriebssystemversionen MS-DOS 4.01 und MS OS/2 Standard Version 1.1 vor, die auch auf dem Deskpro 286e einsetzbar sind. Sowohl die Version 4.01 des Single-user-/Single-tasking-Betriebssystems MS-DOS als auch die Version 1.1 des Single-user-/Multitasking-Betriebssystems OS/2 sollen sich vor allem benutzerfreundlicher präsentieren. Erstere mittels MS-DOS-Shell, einer grafischen Benutzeroberfläche mit Menüs, Fenstertechnik und Piktogrammen. Letztere durch den Presentation Manager (PM), ebenfalls eine leicht zu bedienende grafische Benutzeroberfläche. *MP*

Weitek-Koprozessor auch für 68020/68030-CPU

Nachdem von der Firma Weitek im vergangenen Jahr der erfolgreiche Koprozessor für 80386-CPUs, der 3167, vorgestellt wurde, gibt es jetzt den zu der Prozessorlinie Motorola 68020/68030 passenden 3168 von Weitek. Er enthält einen Gleitkomma-Multiplizierer, eine ALU, eine Divisions-/Radiziereinheit sowie ein Memory-Mapped-Interface zur CPU. So wie der 3167 gegenüber dem Intel-Koprozessor 80387 wesentliche Geschwindigkeitsvorteile besitzt, soll die Steigerung der Verarbeitungsgeschwindigkeit gegenüber dem Motorola-Koprozessor 68882 bis zum 4fachen betragen. Am deutlichsten drücke sich dies bei Anwendungsprogrammen der Bereiche CAD/CAM, Finite-Elemente-Analyse, Grafik sowie Mathematik/Statistik aus. *MP*

Computer kann 3755 chinesische Schriftzeichen erkennen

An der Texterkennung arbeitet seit Anfang 1986 ein Forschungsteam, bestehend aus Sinologen (Chinakundlern) und Informatikern der Technischen Universität Berlin (West). Diesem Forschungsteam ist es gelungen, ein Gerät zu entwickeln, das 3755 chinesische Schriftzeichen liest. In Europa soll es kein vergleichbares Gerät dieser Art geben. In Japan existieren bereits elektronische Lesegeräte und auch in der VR China wird an einer solchen Entwicklung gearbeitet.
In den vergangenen Jahren wurden über 500 Verfahren entwickelt, mit denen chinesische Schriftzeichen in den Computer eingegeben werden können. Da es aber für jedes Schriftzeichen mehrere Dutzend Bedeutungen gibt, ist das Lesen der aus Sinn- und Lautsymbolen zusammengesetzten Wortzeichen sehr schwer. Die chinesische Schrift verfügt insgesamt über einen Stamm von 40 000 bis 60 000 Schriftzeichen. Eine Statistik ergab aber, daß in etwa 98 % aller Texte „nur" 3 755 Schriftzeichen vorkommen.
Das Forschungsteam der TU Berlin (West) arbeitet mit einem als „Schwarzsprung" bezeichneten Verfahren, bei dem jedes gedruckte Schriftzeichen in vier Richtungen abgetastet wird. Mit einer Videokamera werden die auf dem Papier befindlichen Zeichen gelesen (Lesegenauigkeit wird mit 98 % angegeben) und auf dem Bildschirm des Computers wiedergegeben.
Zu Beginn des Forschungsvorhabens, das die Bezeichnung „TECHIS" trägt, wurden ca. 6 000 Schriftzeichen in den Computer eingegeben. Gegenwärtig wird zum Lesen einer Seite eines chinesischen Magazins noch ein ganzer Tag benötigt. Bis 1991 soll aber eine Seite pro Minute gelesen werden können.
Bis zum korrekten Übersetzen, so schätzt das Forschungsteam ein, werden noch mehrere Jahre vergehen.
Quelle: Handelsblatt. – Düsseldorf vom 14. 2. 1989 *Fa*

POWER-RAMs

Mit POWER-RAMs (Parallel Optical Write, Sequential Optical Read) bezeichnen die Forscher der Technischen Universität Delft den von ihnen entwickelten optischen RAM. Dieser neue RAM-Typ soll in Zukunft parallel arbeitenden Prozessorsystemen eine Verarbeitungsgeschwindigkeit von mehr als 10 GFLOPS (Milliarden Gleitkommaoperationen pro Sekunde) ermöglichen.
Jeder RAM-Chip soll aus 4 × 4 optoelektronischen Zellen mit Differentialeingangsstufen bestehen. Die Chips werden einzelnen Prozessoren zugeordnet, die in Delft wiederum zu Arrays von rund 1 000 Prozessoren vereinigt werden. Da die Prozessoren in einem Broadcasting-Netzwerk mit Glasfasern verbunden sind, dienen die POWER-RAMs den Prozessoren als Eingangspufferstufen. Damit soll eine effektivere Parallelverarbeitung als bisher möglich werden, was für die Entwicklung neuer Supercomputer von Interesse ist. *MP*

Schneller RAM mit Pipeline-Technik

Zu einer weiteren Leistungssteigerung herkömmlicher Rechner soll der Prototyp eines neuen, von IBM entwickelten, 128-KBit-SRAM-Speicherchips beitragen. Der Siliziumchip wurde mit der sogenannten Pipeline-Technik hergestellt und soll eine Lese-/Schreibgeschwindigkeit von 6 GBit/s erreichen. Diese Technik ermöglicht, daß das Einlesen der Adressen und das Lesen und Schreiben der Daten parallel über spezielle Signalpfade, mit einer Zykluszeit von

6,5 ns und einer Wortlänge von 32 Bit, ablaufen können. Außerdem können durch die Zwischenspeicherung benötigter Daten aus den relativ langsamen Arbeitsspeichern Wartezyklen für die schnellere Prozessorlogik vermieden werden.

Quelle: Eildienst. – Berlin (1989) 42. – S. 4 Wi

Mini-Bildschirm

Eine ganz besondere Art eines Bildschirmes entwickelte das amerikanische Unternehmen Reflection Technology Inc. Dieser Bildschirm soll die Größe einer Streichholzschachtel haben, 56 Gramm wiegen und wie eine Brille aufgesetzt werden können. Durch eine optische Täuschung sieht man alle Zeichen so, als wären sie in einer Entfernung von ca. 60 cm. Dieser Mini-Bildschirm bietet Platz für 25 Zeilen mit je 80 Zeichen.

Mit dieser Entwicklung beweist das Unternehmen, daß die bislang als undurchbrechbar erschienene Grenze, daß Bildschirme aufgrund der Lesbarkeit eine Mindestgröße nicht unterschreiten dürfen, überwindbar ist.

Quelle: Eildienst. – Berlin (1989) 48. – S. 3 Fa

Neurochips von Mitsubishi

Der Entwicklung eines Neurocomputers ist die Firma Mitsubishi Electric mit der Konstruktion eines optischen Neurochips näher gekommen. Das 32-Neuron-Bauelement besteht aus 32 dreidimensional angeordneten lichtemittierenden Dioden, einer optischen Maske aus 32 × 32 zweidimensional angeordneten Elementen und 32 Fotodiodenelementen auf einem Galliumarsenid-Substrat. Die LEDs und die Fotodioden senden bzw. empfangen Lichtsignale zwischen Neuronen, wobei die optische Maske den Signalaustausch zwischen den Neuronen in Abhängigkeit vorgegebener Steuerinformationen steuert. Der Neurochip kann auf Leiterplatten wie andere LSI-Chips angeordnet werden. Dadurch kann die optische und elektrische Signalverarbeitung vereinigt werden.

Neurocomputer sollen der Verarbeitung großer Informationsmengen dienen, z. B. im Zusammenhang mit Sprach-, Bild- und Handschrifterkennung.

Mit dem Prototyp des Neurocomputers hat Mitsubishi Electric auch ein assoziativ organisiertes Speichersystem demonstriert.

ADN Wi

Zweidimensionaler Draht

An der Entwicklung einer neuen Generation von Bauteilen, die aus quantenelektronischen Strukturen aufgebaut sind, arbeiten Wissenschaftler der Universität von Kalifornien. Dabei ist es ihnen gelungen, ein Geflecht aus sogenannten elektronischen Drähten zu entwickeln, die scheinbar nur aus zwei Dimensionen bestehen (Länge und Breite). Die „Drähte" sind so winzig, daß sechs Millionen von ihnen der Breite eines menschlichen Haares entsprechen.

Den Wissenschaftlern der Universität ist es gelunge, ein Gitter aus Quantendrähten zu entwickeln. Die „Drähte", die Atom für Atom zusammengefügt wurden, bestehen aus Galliumarsenid und die Isolationsschicht aus Aluminiumarsenid. Die Wissenschaftler sind der Ansicht, daß mindestens noch 15 Jahre vergehen werden, bis Produkte mit Quantendrähten auf den Markt kommen werden. Ihre Entwicklung wird aber dazu beitragen, die Miniaturisierung der Chips weiter voranzubringen. Sie sehen z. B. neue Möglichkeiten in der Entwicklung von Laptop-Computern, die mit Blitzlichtbatterien arbeiten. Aus Quantenstrukturen bestehende Laser wären so klein und brauchten so wenig Energie, daß sie auf Satelliten untergebracht werden könnten. Sie sind trotzdem so leistungsfähig, um durch Wasser zu dringen und zu ermöglichen, daß U-Boote miteinander kommunizieren könnten.

Ultrafeine Quantendrähte könnten dazu beitragen, daß Computer noch leistungsfähiger und vor allem auch kleiner werden. Der Grund dafür besteht darin, daß diese Drähte theoretisch den Bewegungsraum der Elektronen einschränken, d. h., daß sie sich wie eine Welle in nur eine Richtung bewegen, und nicht, wie üblich, unregelmäßig im Draht. Demzufolge müßten sich die Elektronen schneller in einem Quantendraht bewegen können als in einem herkömmlichen.

ADN Fa

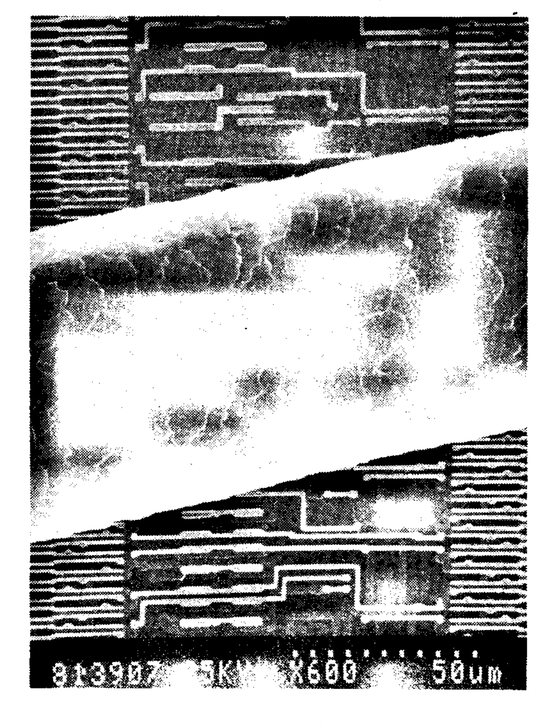
Einblicke in die Mega-Welt ...

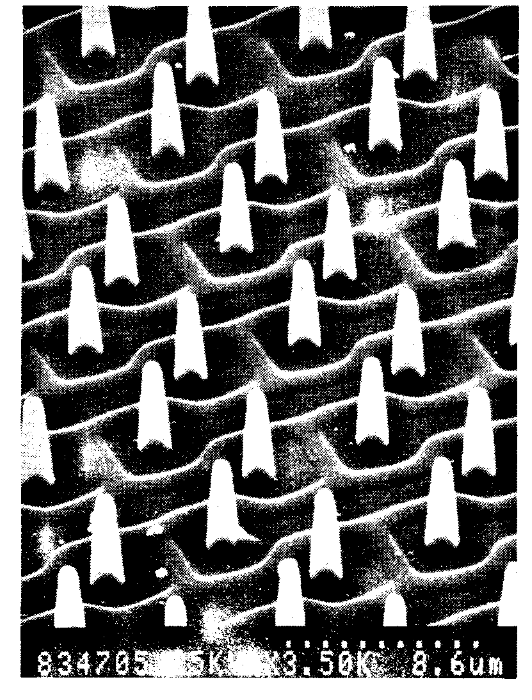
... gewähren die beiden untenstehenden Bilder, die als Aufnahme durch das Rasterelektronenmikroskop entstanden. Sie zeigen Ausschnitte aus dem 4-MBit-DRAM HYB 514000 von Siemens, der noch in diesem Jahr in Serienproduktion gehen soll. Mit diesem Bauelement bewegt man sich bei der Firma erstmals im Bereich der Submicrontechnologie; das heißt, die Abmessungen der Transistoren liegen unterhalb der 1/1000-mm-Marke. Wie fein damit die Strukturen auf dem Zellenfeld werden, verdeutlicht im linken Bild der Größenvergleich mit einem menschlichen Haar. Die Aluminiumleitbahnen wurden zum Teil „gestapelt", so daß deren Gesamtlänge von 25 m auf dem nur 14 × 6,5 mm² großen Chip untergebracht werden konnte. Gegenüber dem 1-MBit-DRAM HYB 511000 wurde die Speicherkapazität zwar vervierfacht, die Chipfläche jedoch nur knapp verdoppelt. Dazu trägt auch das Prinzip der Graben-(Trench-)zelle bei. Bekanntlich besteht bei einem dynamischen RAM jede Speicherzelle aus einer Kombination von Auswahltransistor und Kondensator, der die Information als Ladung speichert. Um Platz in der Waagerechten zu sparen, wurde die Kondensatorfläche in die Tiefe gelegt, also die 3. Dimension genutzt. So gelang es, nicht nur jeweils 4 194 304 Transistoren und Kondensatoren, sondern auch die das Zellenfeld umgebenden Logikschaltungen, insgesamt also 8,9 Millionen Bauelemente, auf der 91-mm²-Chipfläche unterzubringen.

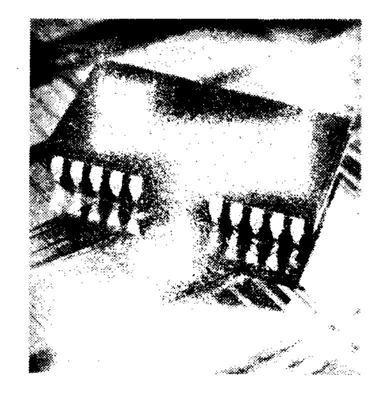
Das rechte Bild zeigt den Blick von unten auf das Zellenfeld des Chips, wobei die Gräben mittels spezieller Präparation sichtbar gemacht wurden und hier wie Kerzenflammen erscheinen.

Gefertigt wird der HYB 514000 in CMOS-Technologie auf 150-mm-Wafern, so daß jeweils 140 Chips erreichbar sind. Die extrem hohen Anforderungen an die Reinheit in diesem Herstellungsprozeß veranschaulicht Siemens mit dem Vergleich eines Fußballfeldes, auf dessen Fläche höchstens ein Schmutzteilchen von der Größe eines Stecknadelkopfes auftreten dürfte. In der Tafel wird diese Entwicklung verdeutlicht. Das Bild ganz unten zeigt den HYB 514000 in der SMD-Version.

MP







Speichergeneration	16 KBit	64 KBit	256 KBit	1 MBit	4 MBit
Speicherbare A4-Schreibmaschinenseiten	1	4	16	64	250
Durchmesser der Si-Scheiben (Produktion) in mm	100	100	125	150	150
Chips pro Scheibe	390	250	220	240	140
Chipfläche in mm²	16	25	45	55	91
Anzahl der Bauelemente	36 500	150 000	565 000	2,2 Mio	8,9 Mio
Zahl der Transistoren	ca. 18 000	27 000	290 000	1,1 Mio	4,7 Mio
Organisation	16 000 × 1	64 000 × 1	256 000 × 1	1 M × 1	4 M × 1
Kleinste Strukturgröße	4,0 µm	2,0 µm	1,5 µm	1,2 µm	0,7 µm
Kritische Defektgröße	1,3 µm	0,8 µm	0,6 µm	0,4 µm	0,2 µm
Technologie	NMOS	NMOS	NMOS	CMOS	CMOS
Zahl der Prozeßschritte	80	80	120	250	ca. 400

Fachtagung KomCom '89

Kommunikations- und Computertechnik Dresden 1989

Vom 14. bis 16. Februar 1989 veranstaltete die Sektion Informationstechnik der Technischen Universität Dresden diese Fachtagung, die von über 700 Teilnehmern aus dem In- und Ausland besucht wurde. Über wissenschaftlich-technische Ergebnisse wurde in den Themengruppen:

- Digitale Vermittlungstechnik
- Mikroprozessor- und Mikrocomputertechnik
- Lokale Kommunikationsnetze
- Funksignalübertragung und -verarbeitung
- Digitale Signalverarbeitung/Signalprozessoren
- Mikroakustische Signalverarbeitung
- Zuverlässigkeit von Kommunikations- und Computersystemen

von mehr als 170 Referenten theoretisch anspruchsvoll, anwendungsbereit bzw. praxisnah berichtet.

Im ersten Plenarvortrag referierte *Lochmann* (IH Berlin) außerordentlich tiefgründig über die weitere Entwicklung der Kommunikationsdienste und -netze. Er arbeitete sehr überzeugend den Einfluß der Hoch- und Schlüsseltechnologien, vor allem der Mikroelektronik und der Lichtwellenleitertechnik auf die Entwicklung bewährter als auch völlig neuer Dienste und Netze heraus. Telefon, Telefax, digitale Netze wurden ebenso behandelt, bewertet und eingeschätzt wie Breitbandnetze und Breitbandkommunikation.

Wertvoll, anregend und treffsicher waren auch die Ausführungen von *Neubert* (TU Dresden) zu den Entwicklungstendenzen in der Computertechnik. Er charakterisierte die bisherige Entwicklung allgemein und beispielhaft, setzte sich mit der heutigen Computertechnik auseinander und versuchte sich auch an einer Prognose, die Zustimmung fand.

Nach diesen beiden Plenarvorträgen begannen die Vorträge zeitgleich in den 7 Themengruppen, wobei nachfolgend nur ein Bericht mit Wertung und Einschätzung zur Themengruppe *Mikroprozessor- und Mikrocomputertechnik* gegeben wird.

Den ersten Hauptvortrag hielt *Giloi* (FIRST, Berlin/West) zum Thema „Super-Mikrocomputer und Mikro-Supercomputer". Ausgehend von den zwei alternativen Technologien für Supercomputer der Höchstgeschwindigkeitstechnologie (VHSIC) und der Höchstintegrationstechnologie (VLSI) ging der Referent auf Architekturen für Supercomputer in enger Beziehung von Theorie und Realisierbarkeit ein. Beeindruckend waren die Ausführungen zur Architektur, technischen Realisierung und Leistungsfähigkeit des Supercomputers „Suprenum", überzeugend führte er aus und wies nach, daß die technologischen Aspekte zu entscheidenden Schlüsselfragen im Forschungs- und Entwicklungsprozeß geworden sind. Die Technologien der Mehrlagenleiterplatten, der Montage von SMD-Bauelementen und der ASICs charakterisieren diese Feststellung hinreichend.

Vielseitig und umfangreich befaßte sich eine große Zahl der Referenten mit der Anwendung hochintegrierter Schaltkreise sowie mit der Logikanalyse. Dazu referierte u. a. *Götze* (TU Dresden) zu einem neuen Steuerungskonzept eines modularen Logik-Analyse-Systems. *Nicklisch* (TU Dresden), *Eichfeld* (VEB Elektronik Gera) und *Rothmann* (TU Dresden) stellten den 100-MHz-Logikanalysator LA 100 vor. *Wolff* (TU Dresden) gingen besonders auf die Leistungsmerkmale des für die Wartung und den Service konzipierten, transportablen Logikanalysators LA 8/20 mit 8 Kanälen und 20 MHz-Abtastfrequenz ein.

Bemerkenswert sind auch die Ergebnisse aus der WPU Rostock zur Erschließung neuer Anwendungen für mikroelektronische Schaltkreise. *Münzer* (WPU Rostock) beispielsweise zeigte und erläuterte Struktur, Aufbau und Funktion des Single-Board-Computers WPU 80286 in einer nachnutzbaren Version. Auch weitere Vorträge aus der WPU Rostock von *Schumacher/Prager* zu Mini-MAP, von *Manzelmann* zu einem Farbgrafiksystem und von Graumann zum Anschluß von Festplattenspeichern an den WPU 80286 fanden Anerkennung und Zuspruch.

Als besonders gelungen und eindrucksvoll ist der Hauptvortrag von *Neumerkel* (TU Berlin, Berlin/West) über Transputer und deren Einsatz in der Spracherkennung und in der Robotik einzuschätzen, weil nicht nur die Transputer schlechthin in Struktur, Vernetzung, Funktion, Aufbau und Einsatz beschrieben wurden, sondern weil eine beachtliche Sorgfalt in die Bestimmung des Leistungsumfanges im Vergleich zu anderen Mikroprozessoren, wie z. B. 80286, gelegt wurde. Die Verkopplung von Transputern über das LINKNET verdient besondere Beachtung. Der Kurzbericht zu den Einsatzerfahrungen mit Transputern rundete den Vortrag sehr gut ab.

Danach stellte *Schlechter* (TU Dresden) seine Ergebnisse zum Einsatz von Transputern in digitalen Vermittlungszentralen dar. Diese Ausführungen ergänzten den Hauptvortrag von *Neumerkel* recht gut, weil sie auch Leistungsgrenzen des Transputers aufzeigten. In diese Gruppe der Vorträge gehört auch der von *Leich* (IfH Zeuthen) gehaltene Vortrag zur Diagnose in einem Multimikroprozessorsystem auf der Basis von Transputern. Leistungsverhalten sowie das Ab- und Zuschalten von Transputern wurden diskutiert.

Eine gute Aufnahme fand auch der Vortrag von *Kämpf* (IfAM Erfurt), der eine gut ausgewählte Kollektion von Ergänzungs- und Erweiterungsbaugruppen für den ESER-PC EC 1834 vorstellte und zur Nachnutzung anbot. Rasterkarte, Busanpassung, Busverlängerung, AD/DA-Leiterkarte u. a. gehörten mit zum Angebot.

Eine Gruppe von vier Referenten befaßte sich mit dem an der AdW der DDR im IIR entwickelten fehlertoleranten Rechnersystem und seinem Einsatz als Paketvermittlungsrechner. Nachdem *Hammer* (AdW der DDR) Struktur, Funktion und Einsatz des fehlertoleranten Rechnersystems übersichtlich und sehr anschaulich dargestellt hatte, referierte *Ahrens* (AdW der DDR) zu den Hardwarekomponenten des Multiprozessorsystems und *Pietsch* (AdW der DDR) zu den Komponenten eines fehlertoleranten Echtzeitbetriebssystems. Schließlich sprach noch *Niepel* (AdW der DDR) zum Entwurf eines Paketvermittlungssystems auf der Basis eines fehlertoleranten Mulitrechner-Multiprozessor-Systems.

Die Referenten stellten insgesamt ein wissenschaftlich-technisches Ergebnis mit beachtlicher technischer und technologischer Reife vor.

Hervorgehoben seien noch zwei Vorträge, in denen besonders Hard- und Softwarelösungen für Prozesse der Bildverarbeitung im Mittelpunkt standen. *Rieken* (TU Karl-Marx-Stadt) referierte eindrucksvoll zu einer Datenflußprozessorbaugruppe, realisiert unter Verwendung des Schaltkreises μPD 7281, für Personalcomputer. *Schmidt* (TH Ilmenau) ging besonders gut auf Struktur, Funktion und Leistungsumfang einer verfügbaren Bildeingabebaugruppe für MMS 16-Rechnersysteme ein.

Nicht unerwähnt bleiben soll der Vortrag von *Meisel* (VEB Carl Zeiss Jena), der über die vorteilhafte Anwendung von FORTH in Rechnern, die für Prüfzwecke eingesetzt werden, berichtete.

Generell kann eingeschätzt werden, daß die KomCom '89 ein anspruchsvolles wissenschaftlich-technisches Niveau dank der guten Qualität der Referate und der Diskussion erreichen konnte. Mit der Vorstellung und partiellen Wertung ausgewählter Vorträge sollten Umfang und Inhalt der Themengruppe „Mikroprozessor- und Mikrocomputertechnik" charakterisiert werden. Kontakte zu den Referenten können über den Veranstalter vermittelt werden.

Die nächste Fachtagung zu dieser Thematik wird 1993 stattfinden.

Prof. Dr. W. Cimander

Fachtagung Computeranimation

Am 9. und 10. Februar 1989 fand an der Technischen Universität „Otto von Guericke" Magdeburg die 2. Fachtagung „Computeranimation" statt. Veranstalter waren die Kammer der Technik, die Sektion Informatik der TU Magdeburg, das Institut für Film, Bild und Ton (ifbt) und die Sektion „Naturwissenschaftlich-Technischer Film" der Nationalen Vereinigung für wissenschaftlichen Film und Fernsehen (NVWF) der DDR.

140 Teilnehmer, darunter auch Vortragende aus der Ungarischen Volksrepublik und aus der BRD, hörten 21 Fachvorträge, die sich vorwiegend auf Live-Demonstrationen stützten. Zusätzlich wurde in den Tagungspausen in 6 Demonstrationen Grafik- und Animationssoftware vorgeführt.

Der Veranstalter stellte dafür unterschiedliche Rechner und Videotechnik zur Verfügung und übertrug die Ausgabe wahlweise auf die Farbmonitore im Tagungsraum, so daß Vortragende und Teilnehmer sehr gute Arbeitsbedingungen vorfanden.

Die überwiegende Zahl der Animationsprogramme wurde auf 8-Bit-Computern entwickelt und demonstriert (KC 85/2, KC 85/4, PC 1715, BC 5120, C 64, Atari, Amiga 500), umfangreiche Simulations- und Animationsprogramme jedoch auch auf 16-Bit-Computern (A 7100, A 7150, Schneider 1512, IBM AT). Über die Animation mit einem CAD-System auf einem 32-Bit-Rechner wurde anhand von Diapositiven berichtet (*Dr. Jetschny*, TU Dresden). Von besonderem Interesse war auch die Rechnerkopplung A 7100–KC 85/3 (*Schreyer*, TU Karl-Marx-Stadt) sowie mehrerer KC 85/3 untereinander (*Schmidt*, TU Magdeburg) zur Übertragung von Bildinformationen und Objektbewegungsdaten.

Gemessen an der Zahl der Vorträge bildete die Simulation und Animation technischer Prozesse mit 9 Vorträgen den Schwerpunkt der Anwendungen und Demonstrationen. 5 Vorträge beschäftigten sich mit der Basisgrafiksoftware zur Bilderzeugung und Animation, 3 Vorträge mit Problemen der Bildgestaltung und bildkünstlerischen Darstellung und 4 Vorträge mit der Anwendung der Animation in der Ausbildung (CAL: Computer Aided Lecture, Teachware, Courseware).

Die Entwicklung der Basisgrafiksoftware stellt gegenwärtig das vordringlichste Problem dar. Für spezielle Fälle wurden ansprechende Lösungen vorgestellt, und zwar

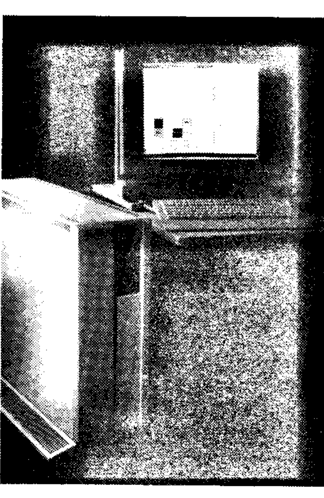
- zur Erzeugung und Manipulation pseudografischer Bilder und Bildfolgen (*Schreyer*, TU Karl-Marx-Stadt)
- zur einfachen Beschreibung geometrischer Objekte und zu deren Manipulation (*Dr. Rennert*, IHS Warnemünde)
- von Schnittstellen zur Grafikprogrammierung unter MS-DOS (*Dr. Hübner*, AdW-ZKI Berlin)
- für die Datenhaltung zur Realisierung schneller Bildwechsel bei begrenzter Bildanzahl (*Ginter*, FSU Jena)
- für Beleuchtungsmodelle und Schattierungstechniken zur realistischen Darstellung von 3D-Szenen (*Haberstroh*, WPU Rostock).

Die Animation von Simulationsresultaten wird durch die Entwicklung von Simulationssoftware für Personalcomputer zunehmend unterstützt. Eine grundsätzliche und systematische Übersicht über Verfahren, Werkzeuge und Arbeitseinrichtungen zur Computeranimation gaben *Prof. Lorenz/Dr. Schulze* (TU Magdeburg) und zur Beschreibung diskreter Systeme für Simulation und Animation *Prof. Grützner* (WPU Rostock). An nutzungsfähiger Simulationssoftware für Personalcomputer, die eine Animation der Simulationsresultate auf unterstem Level (Pseudografik, diskontinuierliche Bewegung von Zeichenketten) unterstützt, wurden Anwendungen in den Programmsyste-

men SIMFOR (*Lohse*, FER Magdeburg/*Dr. Knocke*, TU Magdeburg), SIMPC (*Dr. Schulze*, TUZ Magdeburg), SIMPLEX II (*Dörnhöfer/Germer*, FAU Erlangen-Nürnberg) vorgeführt.
Als Programmiersprachen werden dabei FORTRAN, Turbo-Pascal und C genutzt. Die technischen Anwendungsgebiete der Simulation und Animation bildeten hauptsächlich Materialfluß- und Fertigungssysteme, wobei auch entsprechende Beispiele für die Parallelanimation und die Post-Run-Animation auf KC 85/2 und KC 85/3 zu sehen waren.
Die Animationsanwendungen in der Ausbildung sind ausschließlich für den KC 85 entwickelt worden, wobei die Programmiersprache Basic und z. T. spezielle Assemblerroutinen genutzt werden. Es wurde über Software für die Hochschulbildung informiert (*Dr. Pede*, IFBT); Interessenten erhielten eine Kassette mit Informationen und Programmen von der 1. Tagung Computeranimation 1987 (vgl. Mikroprozessortechnik 2 (1988) 6, Seite 187). Neue Programme mit gelungener Bildgestaltung und Animation wurden zur Unterstützung der Gebiete Statistische Prozeßanalyse (*Dr. Dreyer*, TU Dresden) sowie Sprecherziehung und Rechtschreibübungen (*Bär*, HU Berlin) entwickelt. Die kritisch geführte Diskussion machte deutlich, daß Programmautoren naturgemäß auf ihr jeweiliges Fachproblem fixiert sind und praktikable Empfehlungen zur grafischen (insbesondere farbgrafischen) Gestaltung von Texten, Bildern und Bildfolgen für den Bildschirm vermissen.
Psychologische, werbegrafische und bildkünstlerische Aspekte sind für die wirksame Präsentation der Bildinformationen unerläßlich. Die Wirkung von Farbe, Kontrast, Form und Reihung von Objekten für die visuelle Wahrnehmung dreidimensionaler Szenen in der Natur und für die Darstellung auf dem Flachbildschirm analysierte *Dr. Kolbe* (Hochschule für industrielle Formgestaltung, Halle) eindrucksvoll. Weitere Vorträge behandelten die Anwendung der Videografik und -animation im Fernsehen der DDR (*Müller*, Berlin) und die Einbindung des Computers in den bildkünstlerischen Gestaltungsprozeß (*Prof. Werler*, TU Magdeburg).

Dr. D. Ziems

HANNOVER MESSE CeBIT'89



Von der Firma Symbolics GmbH, die als einer der führenden Anbieter von Symbolverarbeitungssystemen für Entwicklung und Praxis-Einsatz von Anwendungen der KI gilt, wurde die neue Hochleistungs-Workstation XL 400 vorgestellt. Die XL 400 ist gegenüber den bisherigen Symbolics-Workstations der Serie 3600 etwa doppelt so schnell und als offenes Entwicklungssystem ausgelegt.

Die über 3100 Aussteller auf etwa 235000 m^2 Ausstellungfläche sowie über eine halbe Million Besucher belegen, daß die CeBIT eine der größten Fachmessen auf dem Gebiet der Kommunikations- und Computertechnik ist. Die relativ hohe Zahl der Aussteller von außereuropäischen Ländern ist ein weiterer Beweis für diese Einschätzung. Auf der Messe wurden sowohl einzelne Hard- und Software-Erzeugnisse als auch umfangreiche Systemlösungen aus den Bereichen Kommunikations- und Computertechnik vorgestellt.
Gerade der Anwendungsaspekt steht ganz besonders deutlich im Mittelpunkt. Nachfolgend einige Ausführungen zu Entwicklungstendenzen und bemerkenswerten Exponaten.
Control Data Corporation, die im Vorjahr die Fertigstellung des Supercomputers ETA 10 groß aufgemacht hatte, war in diesem Jahr nicht vertreten. Aber es gab weitere Beispiele für die kontinuierliche Entwicklung in diesem Bereich. Auf zwei Systeme sei kurz näher eingegangen.
Die Firma CONVEX (USA) präsentierte Modelle ihrer Serie C, deren Merkmale der Tafel 1 zu entnehmen sind.
Interessant waren ebenfalls Arbeiten zur sogenannten VLIW-Architektur (very long instruction word). Ein Supercomputer der GEI Rechnersysteme GmbH Aachen auf dieser Basis verarbeitet Befehlsworte bis zu einer Länge von etwa 1 KBit auf einer parallelen Prozessorbank. Notwendige Voraussetzung ist ein entsprechender Sprachübersetzer, der nach einer komplizierten Programmanalyse diese Parallelisierung vornimmt. Effizienzverluste entstehen natürlich durch Datenabhängigkeiten und Sprungbefehle. Die Systemfamilie TRACE, die vorgestellt wurde, bietet Leistungen im Bereich von 10 bis 120 MFLOPS.
Die 32-Bit-Minicomputer stellen ein großes Segment im Gesamtspektrum der Computer dar. Verschiedene Firmen bieten unterschiedlich leistungsstarke Systeme an, so daß für den jeweiligen Einsatzfall geeignete Computer auswählbar sind. Durch Kopplungen (Clusterbildung) lassen sich Leistungssteigerungen bei wachsenden Anforderungen erreichen.
Die Workstations und Personalcomputer waren, wie in den letzten Jahren, außerordentlich stark vertreten. Dem Besucher wird eine unübersehbare Vielfalt von Arbeitsplatz- und Personalcomputern geboten. Die Grenze zwischen diesen beiden Klassen wird zunehmend verwischt. Die „klassischen" Workstations (z. B. Sun-Serie 3/60, Apollo) sind mit Prozessoren 68020 bzw. 68030 der Firma Motorola ausgerüstet. Vielfach werden sie ergänzt durch Spezial- und Koprozessoren zur beschleunigten Ausführung von speziellen Aufgaben. Im oberen Leistungsbereich schließen sich Systeme an, die mit Spezialprozessoren ausgerüstet sind. Beispielsweise setzt Sun im System Sun-4 einen RISC-Prozessor eigener Entwicklung ein. Das Konzept wird als Scalable Processor Architecture (SPARC) bezeichnet. Man sagt, daß Sun mit diesem Konzept Anfang der 90er Jahre bis zu 100 MIPS im Workstationbereich erreichen will.
Wie bereits ausgeführt, verwischt sich die Grenze zwischen diesen beiden Klassen kontinuierlich. Vergleicht man beispielsweise ein Top-Modell des PC-Bereiches mit einem unteren aus der Workstation-Klasse (Tafel 2), so zeigt sich die Richtigkeit dieser These. (Bei einigen Werten wurden Annahmen gemacht, um eine Vergleichbarkeit zu gewährleisten. Beispielsweise sind die Speicher weiter ausbaubar). Die Personalcomputer sind in der Hauptsache mit den Intel-Prozessoren 80286 (16 Bit) bzw. 80386 (32 Bit) ausgerüstet. Hauptspeichergrößen bewegen sich im MByte-Bereich, eine Aufrüstbarkeit mit weiteren Speichermoduln oder Koprozessoren ist gegeben. Die Festplatten- und Disketten-Laufwerke des 3,5"-Formats haben jetzt bereits einen Marktanteil von über 20 % erreicht und dürften damit die 5,25"-Diskette bald überflügeln. Trotzdem vollzieht sich dieser Übergang langsamer als ursprünglich angenommen. Bei im PC befindlichen Festplatten fängt die Kapazität bei 20 MByte an und geht bis über 100 MByte hinaus. Die Diskette liegt im Bereich von 800 KByte bis einige MByte. Bei den Druckern machen trotz gestiegener Anforderungen durch das Desktop-Publishing die Mosaikdrucker (vor allem preiswerte 24-Nadeldrucker) immer noch das Rennen. Der Laserdrucker, vor allem als Auftischgerät, verzeichnet jedoch ein größer werdendes Einführungstempo. Bei Bildschirmen steigen das



Das neue Flaggschiff von Tulip ist der AT 386 mit 25-MHz-Prozessor. Der 4-MByte-RAM kann auf der Grundplatine auf 13 MByte erweitert werden; ein 32-KByte-Cache beschleunigt den Zugriff. Als Besonderheit werden alle Funktionen des AT von dem bei Tulip entwickelten Sicherheitsmechanismus, dem System Control Manager, überwacht und gesteuert.

Auflösungsvermögen und der Farbanteil weiter an.
Bei den sogenannten Laptop-Computern (Gewicht 4...8 kg) werden nunmehr durchweg die gleichen Prozessoren, Speicher (auch Massenspeicher, z. B. Harddisk!) und Betriebssysteme eingesetzt wie bei PCs. Als der weltweit erste Laptop (Firmenangabe) mit einem Farbdisplay wurde von Hitachi der HL 400 C vorgestellt. Eingesetzt wurde ein 6,3"-Flüssigkristall-Bildschirm auf Basis der Thin-Film-Transistor-Technologie (TFT), wobei die Dünnfilmtransistoren direkt auf der Glasoberfläche des Flüssigkristall-Bildschirms aufgebracht werden (siehe MP 4/89, S. 124).
Einige weitere interessante Besonderheiten:
- Bei einigen Firmen wird das Konzept einer sogenannten diskless workstation diskutiert (z. B. HP). Der ohne Zweifel notwendige Massenspeicher wird durch Vernetzung aus zentralisierten Ressourcen (Fileserver) bereitgestellt. Nicht zuletzt würden sich damit Probleme der Datensicherheit (Virenbefall!) besser lösen lassen.
- Analog zu dem früheren Intel-„Gespann" 8086 und 8088 vollzieht sich heute das gleiche noch einmal im 32-Bit-Bereich. Der 80386 wird in einer sogenannten SX-Version als 80386 SX (bzw. 80387 SX) mit intern 32 und extern 16 Bit gefertigt. Vermutlich spielen ökonomische Gründe eine wesentliche Rolle bei dieser Vorgehensweise (s. MP 9/88, S. 278).
- Die Firma DEC hat mit ihrer ersten RISC-Workstation (DECstation 3100) die 1000-$-Marke für 1 MIPS unterboten (15 MIPS für 12000 $). Überhaupt hat das RISC-Konzept viel Bewegung im Workstationbereich (und auch darüber hinaus) gebracht. Prognosen sprechen davon, daß 1992 bereits 50 % aller 32-Bit-Computer nach der RISC-Struktur realisiert sind. Dies scheint fraglich, da viele die Leistung steigernden Merkmale des RISC-Konzeptes auch für den CISC-Computer relevant sind und sicherlich auch genutzt werden.

Die im Bericht über die CeBIT'88 bereits angedeuteten Tendenzen bei den peripheren Geräten haben sich weiter ausgeprägt. Deshalb nur einige kurze Bemerkungen.
Die Druckerpalette, die für den PC- und Workstationbereich vorgesehen ist, zeichnet sich durch eine sehr große Vielfalt an einzelnen Erzeugnissen aus. Für die einzelnen Druckertypen ist festzustellen, daß
– Typenraddrucker als Schönschreibdrucker faktisch verschwunden sind. Diese Aufgabe übernehmen ohne Qualitätsabstriche Mosaik- und Laserdrucker.
– Mosaikdrucker den Hauptteil des Angebotes ausmachen. Dominant sind dabei Druckköpfe mit 24 Nadeln. Diese Technik ist bei weitem noch nicht ausgereizt. Weitere Leistungsmerkmale wie Code-Karten für die Druckeranpassung (s. 2 und 3. Umschlagseite), rationelles Handling beim Einzelblatteinzug sowie beim

Übergang von Einzelblatt- zur Endlospapierverarbeitung zeichnen neue Geräte aus. Diese Drucker bieten vereinzelt auch farbige Ausgaben.
– dem Schritt, den EPSON mit dem 48-Nadel-Drucker TLQ-4800 zur CeBIT'88 ging, bisher niemand gefolgt ist.
– Tintenstrahldrucker hochkomplizierte Druckköpfe mit 24 und 48 Düsen nutzen und damit eine exzellente Druckqualität liefern. Auch farbige Ausgaben sind mit derartigen Druckern möglich. Diese Drucktechnik erfüllt höchste Anforderungen für den Einsatz im Bereich CAD.
– der Laserdrucker vor allem als Auftischgerät langfristig sicherlich auch den Mosaikdrucker ablösen wird. Die billigsten Arbeitsplatzdrucker dieser Technik kosten im Vergleich zur CeBIT'88 wohl nur noch die Hälfte.
– ein neues Druckprinzip eventuell für eine neue Gerätefamilie sorgt. Die Firma Qume präsentiert als neues Druckprinzip die Liquid Crystal Shutter (LCS)-Technologie, die das Grundprinzip und alle Ausgabemöglichkeiten des Laserdruckers zum Preis eines Low-cost-Matrixdruckers bietet. Mit 6 Seiten/Minute und 300 × 300 Punkten je Zoll werden bemerkenswerte Leistungen erreicht (s. auch MP 4/89, S. 124).
– nach wie vor große leistungsstarke Drucker (Zeilendrucker, Laserdrukker) für große Computer und damit schnelle Ausgaben existieren.

Vor allem japanische Firmen bieten bei den Massenspeichern ein breites Angebot an Disketten- und Festplatten-Laufwerken. Das Angebot der Floppy-Laufwerke ist im Prinzip unverändert. Laufwerke mit Senkrechtaufzeichnung und entsprechend großer Kapazität haben noch keinen Durchbruch erzielt. Auch bei den Harddisks ist die Situation ähnlich wie im Vorjahr. 5,25"-Laufwerke bieten 25 bis 780 MByte, 3,5"-Laufwerke 50 bis 320 MByte; die Zugriffszeiten liegen um 20 ms und die Datenübertragungsraten bei 2 MByte/s (4 MByte/s im Synchronbetrieb). Bis zu 1,5 GByte bieten die „großen" 9"-Hard-Disk-Drives. Dazu kommen noch 5,25"-CD-ROM-Drives mit einer Kapazität um 500 MByte und mehr. Die letzteren sind vor allem für die Massendatenspeicherung in Informationssystemen einsetzbar. Das mehrfache Lesen und Schreiben ist noch nicht in dem Lösungsstadium, das eine entsprechende Gerätepalette berechtigt.

Bei Bildschirmen ist eine weitere Vervollkommnung der Leistungsmerkmale hinsichtlich Auflösungsvermögen und Farbqualität zu verzeichnen. Schirmgrößen sind 13" und 14", und das Auflösungsvermögen liegt im Bereich 800 × 400 Pixel, für den Einsatz in leistungsfähigen Workstations gehen diese Werte auf 19" und 1 600 × 1 280 Pixel. In solchen Fällen erreicht die Bandbreite des Displays 200 MHz.

Auf dem Sektor der Basissoftware ist zunächst festzustellen, daß das MS-DOS nach wie vor einen großen Raum einnimmt. Das Vordringen von MS-OS/2 ist unverkennbar, jedoch ist gegenwärtig noch eine gewisse abwartende Haltung dazu festzustellen. Der hohe Speicherplatzbedarf wirkt anscheinend doch etwas hemmend auf eine schnelle Einführung. Trotzdem laufen durchaus bereits Software-Produkte auch unter diesem System. Die Ankündigung des OS/2-LAN-Managers von Microsoft eröffnet ein abgestimmtes Konzept zur Vernetzung dieser PCs. Das MS-DOS, das gegenwärtig nach wie vor **das** Betriebssystem für die PCs ist, bleibt in seinen Leistungsmerkmalen auch nicht stehen. Es werden nunmehr auch Funktionen integriert, die für den Nachfolger OS/2 typisch sind. Spezialbetriebssysteme (z. B. Windows/386, VM/386), die seinerzeit als „Lückenschließer" für den PC mit der 80386-CPU fungierten, haben aber trotz OS/2 ihre Postionen weiter ausgebaut. Im oberen Bereich der PCs und der Workstations haben UNIX bzw. Derivate einen festen Platz, der faktisch ungefährdet ist. Darüber hinaus existieren natürlich die „klassischen" Betriebssysteme von DEC, IMB, CDC, SIEMENS u. a. für die entsprechenden Computer.

Auf die unübersehbare Vielfalt der Anwendersoftware kann hier nicht eingegangen werden. Es ist davon auszugehen, daß für alle denkbaren Einsatzgebiete im Bereich der großen Unternehmen, der mittelständischen Industrie, der Handwerksbetriebe und der Privatsphäre Software-Systeme zur Rationalisierung der informationellen Prozese existieren. Anwendersoftware war auch wieder ein ausgesprochener Schwerpunkt der CeBIT'89. Weit über 100 neue Anbieter zu diesem Komplex sind Beweis dieser Einschätzung.

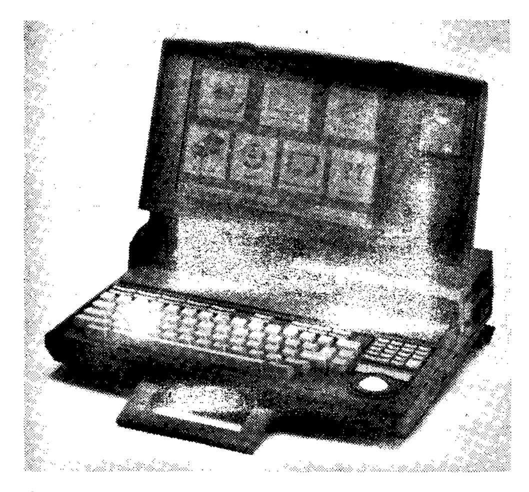
Die Kommunikationstechnik, die die zweite große Säule der Messe darstellt, stand in diesem Jahr unter dem Schwerpunkt des Beginns der flächendeckenden Einführung von ISDN durch die Bundespost. Im Kommunikationsbereich ist festzustellen, daß sich der PC, um bestimmte Komponenten ergänzt, zunehmend mehr als Kommunikationsendgerät etabliert. Diese These beweisen Zusatzkarten für den Anschluß an das Telex-, Teletex oder Bildschirmtext-Netz, als Fax-Gerät, als Datenendstelle am ISDN-D-Kanal u. v. a. m.

Auf dem Gebiet der Rechnernetze machte die CeBIT'89 deutlich, daß die Verkopplungsmöglichkeiten weiter zunehmen. Erkennbar wird das durch folgende Tatsachen:
– Möglichkeiten zur Kopplung unterschiedlicher Netze (z. B. Ethernet und Token)
– Anschlußmöglichkeiten von Geräten unterschiedlichster Hersteller
– weitere Ausgestaltung des Software-Bereiches für diesen Zweck
– ISDN als Basisdienst bietet an verschiedenen Schnittstellen neuartige Möglichkeiten zur Datenkommunikation, die sicherlich bald auch für die Rechnerkommunikation erschlossen werden.

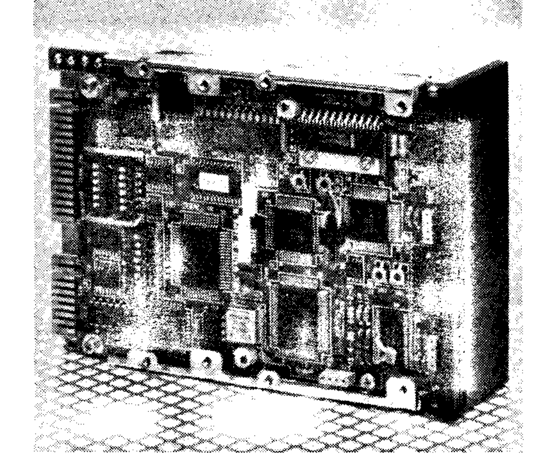
Bei den lokalen Netzen ist bei aller Standardisierung eine weitere Zunahme der Erzeugnisvielfalt zu verzeichnen. Dominierende Konzepte sind dabei nach wie vor das CSMA/CD- und das Token-Verfahren, die für die Einsatzbereiche Büroautomatisierung und flexible Automatisierung (Echtzeitverhalten!) klar favorisiert sind. Für geeignete Anwendungen empfehlen sich „Dünnkabel-Lösungen" als billigere Varianten für ein Ethernet (Verzicht auf den Transceiver). Endgeräte unterschiedlichster Leistungsmerkmale können immer problemloser an Rechnernetze angeschlossen werden, da die Hersteller konsequenter die internationalen Standards beachten. Das gleiche gilt für Lösungen zur Verkopplung von Rechnernetzen durch Gateways. Hierfür sprechen Exponate von IBM und anderen Herstellern (Kopplung von Ethernet- und Token-Netzen, Anschlußmöglichkeiten zur „IBM-3270-Welt") und zur Verbindung mit ISDN- und Paket-Netzen. Auf dem Software-Sektor ist festzustellen, daß Vernetzungskonzepte auf der Basis der „alten" TCP/IP-Protokolle sowie neuer OSI-gerechter Protokolle zu verzeichnen sind.

Zusammenfassend kann festgehalten werden, daß neben den immer zu findenden interessanten technischen Lösungen vor allem der Schwerpunkt auf einem rationellen Einsatz dieser Technik liegt. Die Nutzung von Geräten und Systemen der Computer- und Kommunikationstechnik zur Rationalisierung der vielfältigsten informationellen Prozesse entscheidet schließlich und letztlich über die erzielbare Wirkung.

Prof. Dr. P. Neubert



Atari rundete jetzt seine Angebotspalette nach unten und oben ab: Vom MS-DOS-kompatiblen Leichtgewicht PC Folio bis zur Transputer-Workstation ATW reicht nunmehr das Spektrum. Das hier gezeigte Modell STACY ist der neue ST von Atari – als Laptop. Ausgestattet ist er mit 8-MHz-68000-Prozessor, 1 MByte RAM, einem 3,5"-Diskettenlaufwerk und als Besonderheit mit integriertem Trackball anstelle der sonst üblichen Maus.



NEC zeigte neben bewährten und neuen Druckern, MultiSync-Monitoren und Fernkopierern auch seine Festplattenlaufwerke. Unter anderem das abgebildete 3,5"-Winchesterlaufwerk D 3661 mit 134,5 MByte unformatierter Kapazität und einer mittleren Zugriffszeit von 23 ms sowie einer ESDI-Schnittstelle.

Tafel 1

Modell	C120	C201	C210	C220	C240
Prozessoren	1	1	1	2	4
Prozessorzyklus (ns)	100	55	40	40	40
virtueller HS (GByte)	4	4	4	4	4
max. HS (GByte)	1	2	2	2	2
Logik	CMOS	CMOS +ECL	CMOS +ECL	CMOS +ECL	CMOS +ECL
Durchsatz (MFLOPS)	20	35	50	100	200
Kühlung	Luft	Luft	Luft	Luft	Luft
Leistungsaufnahme (kW)	2,5	5,7	9,3	12,4	18,6

Tafel 2

Modell	IMB PS/2 Modell 80	Sun 386i
Prozessor	I 80386	I 80386
Takt (MHz)	20	20
Display	Farbe, 12"	Farbe, 14"
Auflösung	640 × 480	1 152 × 900
HS (MByte)	4	4
Disk (MByte)	80	80
Preis (TDM)	20	32

Fortsetzung von 2. Umschlagseite

Mit dem neuen 24-Nadel-Drucker SL-230 AI stellte Seikosha zur CeBIT '89 und zur Leipziger Frühjahrsmesse 1989 das Modell einer neuen Druckerfamilie vor, das entsprechend den Markterfordernissen entwickelt wurde und einige weit über diese Klasse hinausgehende Funktionen beinhaltet (Bild 1). Mit einem neu entwikkelten Druckkopf mit paralleler Nadelanordnung bietet der SL-230 AI eine Druckgeschwindigkeit von 230 Zeichen/Sekunde im EDV-Modus und 77 Zeichen/Sekunde im Korrespondenz-Modus bei einer Schriftbreite von 10 Zeichen/Zoll. Dies entspricht bei einer Zeilenbreite von 136 Zeichen einer Durchsatzrate von etwa 83 Zeilen/Minute im EDV-Modus und 32 Zeilen/Minute im Korrespondenz-Modus.

Das Gerät ist serienmäßig mit 9 Schriftarten ausgerüstet (Courier, Gothic Orator, Prestige Elite, Script, OCR-A, OCR-B, S. Roman und S. Helvetica), und jede Schriftart läßt sich mit unterschiedlichen Attributen darstellen.

Die Zeichenabstände sind auswählbar zwischen 10, 12, 15, 17.1 und 20 Zeichen/Zoll. Die Auflösung von maximal 360 × 360 Punkten/Zoll bietet beste Grafikqualität.

Bereits standardmäßig ist das Gerät parallel mit der Centronics- sowie mit einer RS 232 C-Schnittstelle ausgerüstet und beinhaltet die Emulationen IBM Proprinter XL 24 und EPSON LQ-1050.

Als Zeichensätze sind die ASCII- und IBM-Zeichensätze vorhanden, wobei 17 Ländervarianten ausgewählt werden können. Der Speicher hat eine Größe von 64 KByte und ist um zusätzliche 64 KByte erweiterbar.

Neben einem Push-Traktor mit Papierpark-Funktion besitzt das Gerät einen halbautomatischen Einzelblatteinzug, und als Option kann eine vollautomatische Zweischacht-Einzelblattzuführung sehr leicht nachgerüstet werden. Umfangreiche Papiertransportfunktionen wie 1''-Cut, Visible Linefeed, TOF Set und anderes können direkt vom Bedienpaneel aus angewählt werden.

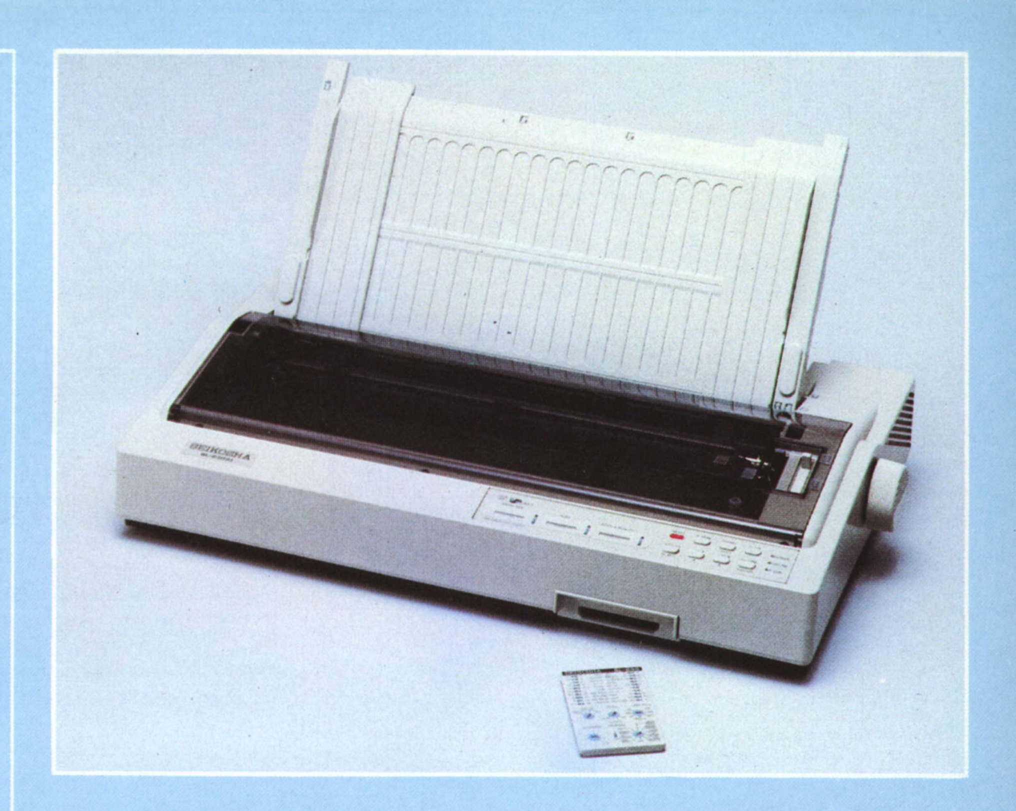
Durch eine **Programmkarte** kann auf sehr einfache Art und Weise der gewünschte Wert für Prozedur, Ländervariante, Papiererkennung, Pufferspeicher und weitere Funktionen gesetzt werden (Bild 2). Die Werte werden mit dem Einschalten des Druckers übernommen und sind gespeichert.

Bei Verwendung des Druckers für verschiedene Applikationen kann jeder Anwender seine persönliche Programmkarte benutzen und jederzeit leicht modifizieren. Die bisher üblichen Einstellungen am Drucker entfallen.

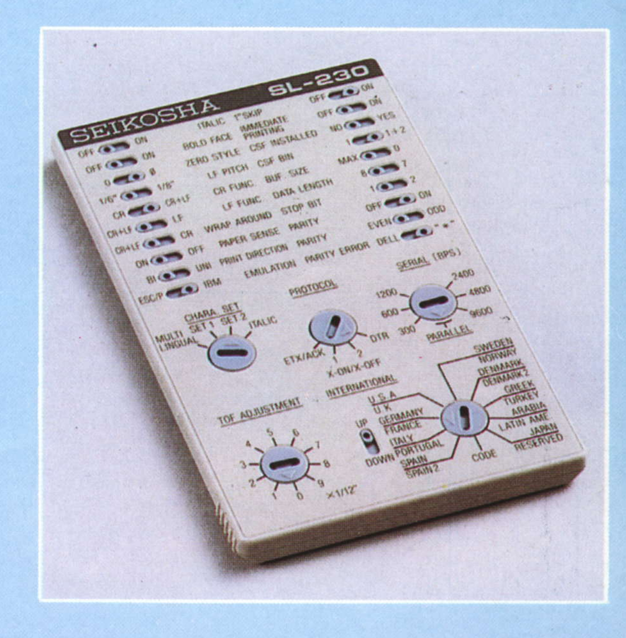
Eine Programmkarte gehört zur Standardausrüstung, weitere sind erhältlich.

Ein Farbband kann für den Druck von 5 Millionen Zeichen genutzt werden.

Der SEIKOSHA SL–230 AI soll seit April 1989 lieferbar sein.



Erster Drucker mit Programmkarte

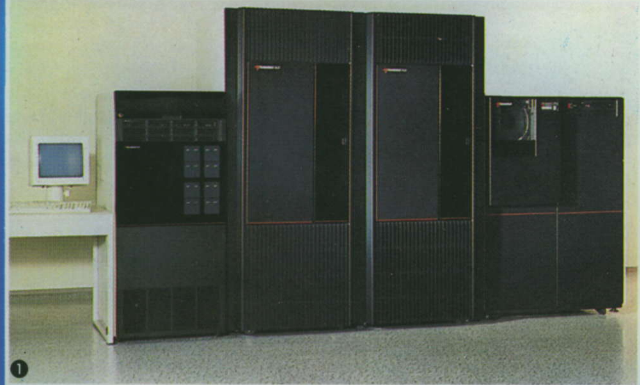


Tafel 1 Technische Daten SL–230 AI

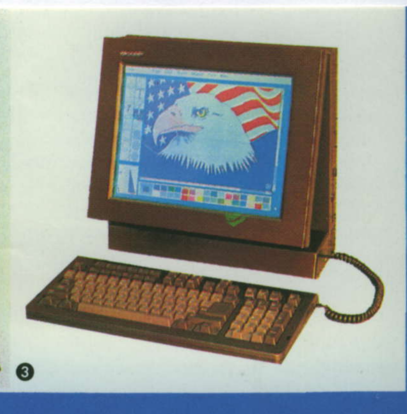
Druckmethode	24-Nadel-Matrix, bidirektional
Druckgeschwindigkeit (Zeichen/Sekunde)	230 in EDV-Qualität, 77 in Schönschrift
Zeilenvorschub (Zoll)	1/6, 1/8, n/60, n/72, n/180, n/360
Zeichensätze	18 internationale, 2 IBM
Zeichenzahl	185 Zeichen und Symbole, 49 internationale Zeichen
Emulationen	ESC/P, IBM Proprinter
Papierformate (Endlospapier)	4–16 Zoll Breite 8–22 Zoll Länge
Papiervorschub	Walze mit halbautomatischem Papiereinzug, Schubtraktor, Papierpark-Funktion
Interface	Centronics parallel, RS 232 C
Druckpuffer	5 KByte, Option 64 KByte
Besondere Merkmale	Komfortables Bedienfeld, hinterlegbare Funktionen über Programmkarte, Quiet Mode
Optionen	Vollautomatische Einzelblattzuführung (1+2 Schacht) RS 232 C-Schnittstelle

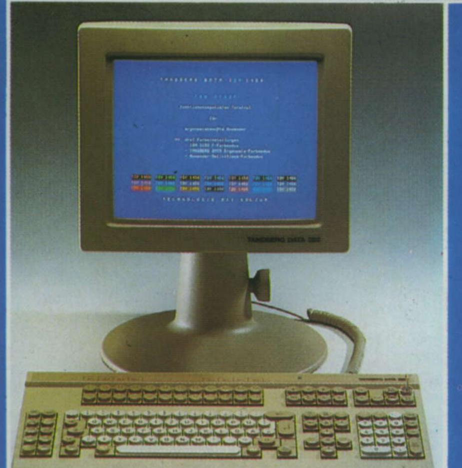
ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 6
S. 161–192

Neuheiten auf der Ce





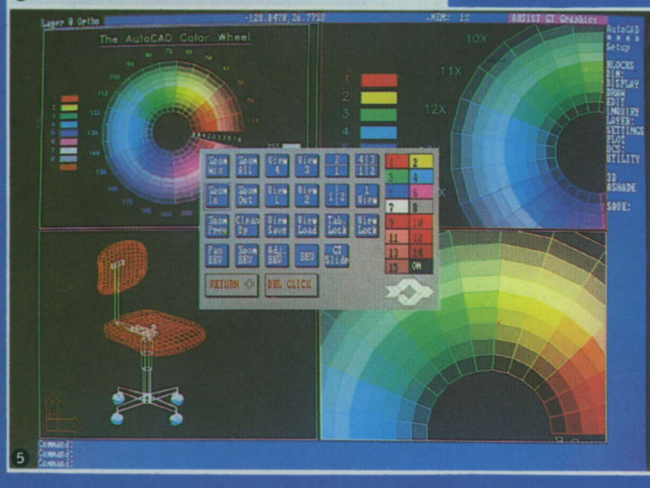


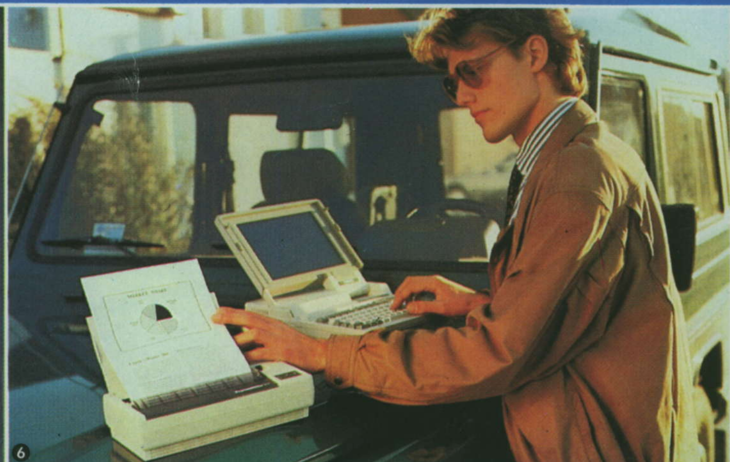


❶ Ausfallgeschützter Parallelrechner
Der **Tandem NonStop VLX** ist der leistungsstärkste der modular strukturierten und ausfallgeschützten Parallelrechnerfamilie von Tandem Computers. Er enthält maximal 32 Prozessoren mit je 16 MByte Hauptspeicher. Alle Modelle benutzen das Betriebssystem Guardian 90.

❷ 32-Bit-Datenbus mit EISA-Merkmalen
Zeniths neuer PC **Z-386/25** ist ein 386-AT mit 25 MHz Taktfrequenz, 2 bis 32 MByte RAM, 64 KByte Cache-Speicher und einer 70- oder 150-MByte-Harddisk. Er besitzt eine VGA-Grafikkarte und arbeitet mit MS-DOS 3.3 plus sowie optional mit OS/2 oder XENIX V.5. Von den 7 Slots haben 4 eine Datenbreite von 32 Bit; ihre Signale sollen eine EISA-Untermenge darstellen. Damit wäre der Z-386/25 der erste EISA-kompatible PC. Er soll seit Mai lieferbar sein.

❸ Portable mit 14''-Farb-LC-Bildschirm
Der erste 386-Portable mit 14''-Farb-LC-Bildschirm ist der **Sharp Farb-Portable 386.** Sein 80386-Prozessor wird mit 20 MHz getaktet, er besitzt eine 40-MByte-Festplatte und einen RAM von 2 bis 8 MByte. Der hintergrundbeleuchtete Bildschirm ist VGA-fähig und kann bis zu 512 Farben darstellen (jede Grundfarbe besitzt bis zu 8 Farbabstufungen). Als Betriebssysteme stehen MS-DOS und OS/2 zur Verfügung. Erste Geräte werden für den Herbst erwartet.





❹ Farbterminal mit Pastelltönen
Im IBM 3192 F-kompatiblen 14''-Terminal **TDV 1450** hat Tandberg Data sein ergonomisches Farbkonzept fortgesetzt. Die Verwendung von Pastelltönen gewährleistet eine optimale auf das menschliche Auge abgestimmte Farbdarstellung. Acht Vorder- und acht Hintergrundfarben können außerdem aus einem Angebot von etwa 256 000 Farben ausgewählt werden. Die Bildfrequenz beträgt bis zu 71 Hz; Auflösungen bis 792 × 497 Pixel sind möglich.

❺ Treiber für AutoCAD Version 10
Der **ARTIST GT**-Treiber von Control Systems für AutoCAD Version 10 bietet für AT- und PS/2-PCs neben hochauflösender Grafik unter anderem zusätzlich: separate Display-Listen für mehrere Darstellungsfelder, Geschwindigkeitsparität mit Hochleistungsrechnern, frei plazierbares Symbolmenü, 6 Zoom-Optionen, Menüsteuerung der Farbebenen, (umschaltbare) Darstellung auf 2 oder 4 Bildschirmfenstern sowie in der Größe veränderbares, frei plazierbares Vogelperspektiv-Fenster.

❻ Portable-Drucker mit Schönschreibqualität
Toshiba bot mit dem **ExpressWriter 301** den ersten netzunabhängigen, transportablen Drucker mit der hohen Auflösung von 360 × 360 Punkten pro Zoll. Der 1,9 kg schwere und 310 × 140 × 74 mm^3 große Drucker besitzt einen Druckkopf mit 24 Thermoelementen für Kopier- und Thermopapier sowie direkt für Folien, 4 residente Schriftarten (andere können nachgeladen werden) und als Interface eine Centronics-Schnittstelle. Die Akkus erlauben eine Druckdauer von knapp einer Stunde bzw. eine Einschaltdauer (ohne Druck) von bis zu 5 ½ Stunden.

Lesen Sie hierzu auch unseren CeBIT-Bericht.



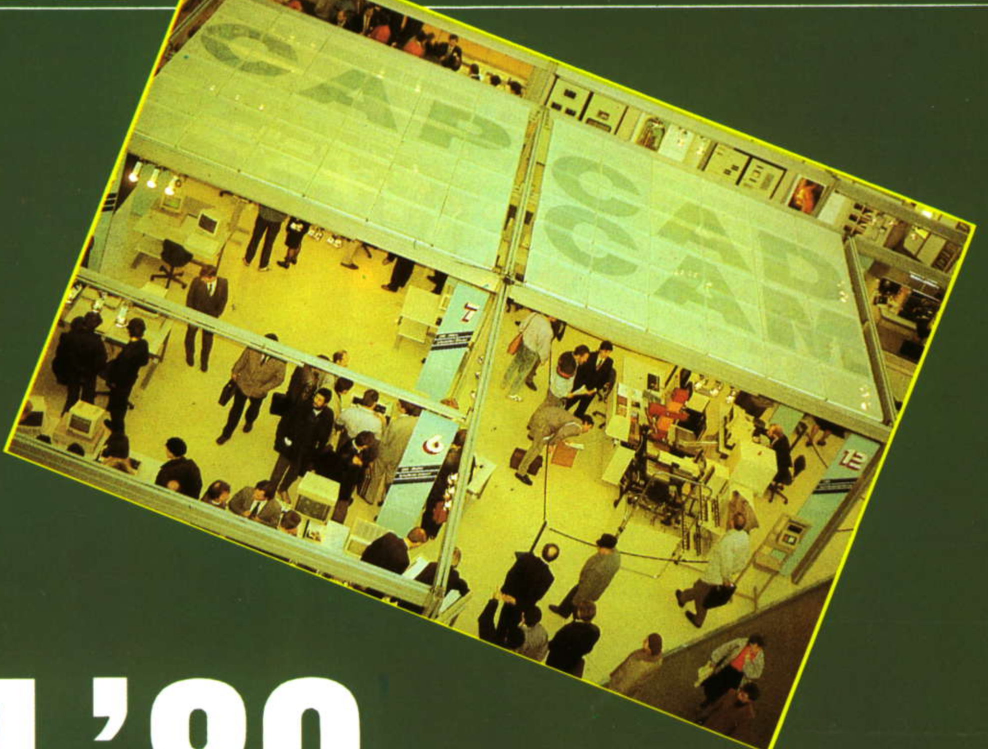
ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 7 · 1989

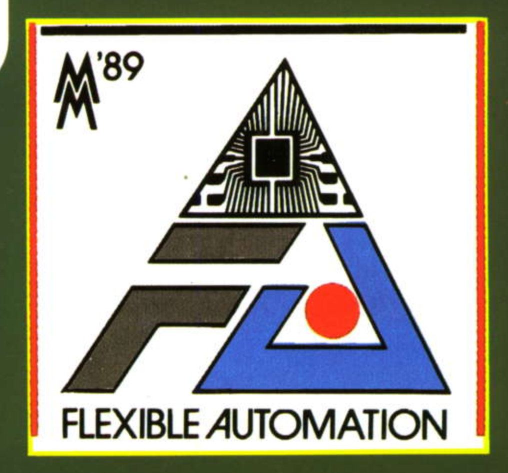
Mikroprozessortechnik

VEB Verlag Technik Berlin
ISSN 0232 – 2892



LFM '89

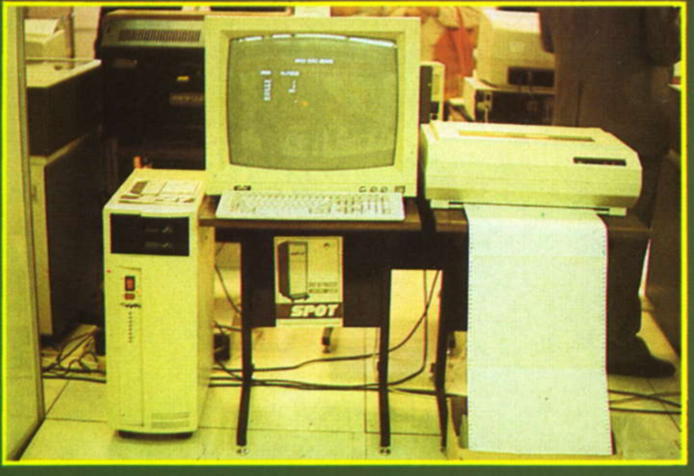




außerdem:

Logikanalyse des 8086/88

Logischer Entwurf von Datenbanken



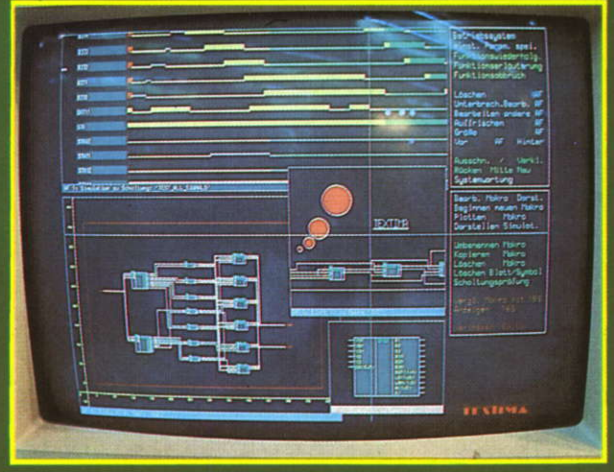
1 *Prozeßrechnersystem Minispot*



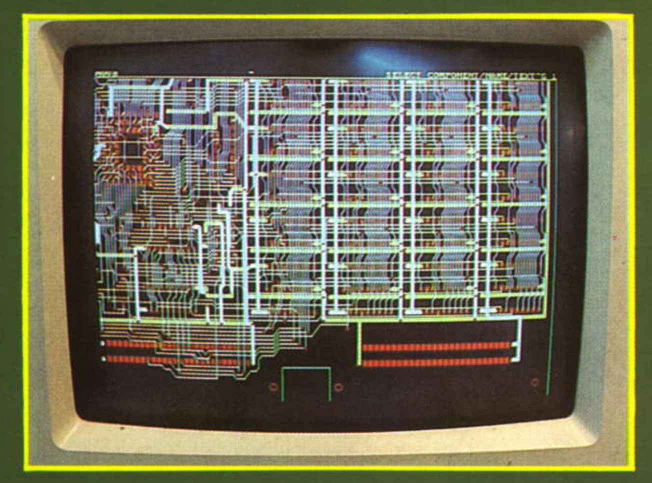
2 *Schaltkreisentwurfssystem ARCHIMEDES*



3 *Terminal K 8919.11 mit Programmsystem UMSCHA*



4 *Grafischer Schaltungseditor TEXgraf*



5 *Leiterplattenentwurfssystem PCLES*



6 *CAD/CAM/CAE-Workstation datagraph Starstation*



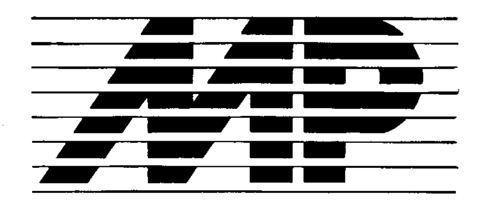
7 *Tektronix-Farbgrafikterminal Tek GS 4211*



8 *Siemens CADIS*

Lesen Sie hierzu unseren Bericht von der LFM '89

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 7 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 287 00, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2 87 02 03), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Erfüllungsort und Gerichtsstand Berlin-Mitte. Der Verlag behält sich alle Rechte an den von ihm veröffentlichten Aufsätzen und Abbildungen, auch das der Übersetzung in fremde Sprachen, vor. Auszüge, Referate und Besprechungen sind nur mit voller Quellenangabe zulässig.

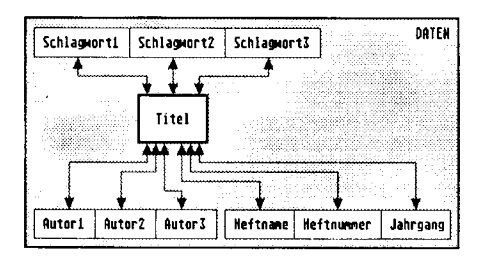
Redaktionsschluß 10. Mai 1989

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

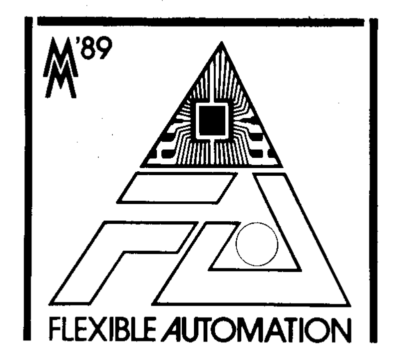
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustředni Expedicia a Dovoz Tisku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trưng, Hà Nội; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Unter dem Titel „Logischer Entwurf von Datenbanken" werden auf der **Seite 195** die Probleme beim logischen Strukturieren von Datenbanken dargestellt. Es wird gezeigt, wie Datenelemente so zu Datengruppen zusammengefaßt werden können, daß die entstehenden Datenstrukturen stabil und nicht anfällig gegenüber Änderungen sind.



Mit der Entwicklung leistungsfähiger Personalcomputer und des Laserdruckers zum Tischgerät wurde ein neues Einsatzgebiet für PCs möglich: das Desktop Publishing (Publizieren auf dem Schreibtisch).
Zum DTP gehören Eingabe und Verarbeitung von Text und Bildern sowie die Ausgabe von gestalteten Seiten für Dokumentationen, Prospekte und ähnliches auf hochauflösenden Druckern. Näheres zu diesem relativ neuen Einsatzgebiet finden Sie auf der **Seite 199.**



Der erste Teil unseres Berichtes von der LFM '89 auf der **Seite 221** konzentriert sich auf das Gebiet des rechnergestützten Entwurfs sowie auf die Computertechnik, die nicht CAD/CAM-spezifisch ist. Der zweite Teil in MP 8/89 wird über neue Bauelemente und periphere Hardware sowie über weitere Softwareangebote informieren.

Inhalt

Vorschau

Für MP 8/1989 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Kopplung von Rechnern unter UNIX
- Diskrete Fouriertransformation und digitale Signalverarbeitung

Erweiterung der Zusammenarbeit in der Elektronik beraten

Die Realisierung des Komplexprogramms des wissenschaftlich-technischen Fortschritts der RGW-Mitgliedsländer bis zum Jahre 2000 stand im Mittelpunkt der 3. Tagung des RGW-Komitees für Zusammenarbeit auf dem Gebiet der Elektronisierung Anfang April in Prag. Dabei wurden entsprechend dem Auftrag der 44. Ratstagung des RGW die erforderlichen Schritte zur weiteren Verbesserung der Deckung des Bedarfs an Bauelementen und anderen mikroelektronischen Erzeugnissen beraten. Außerdem stimmten die Teilnehmer Maßnahmen zur Beschleunigung von Entwicklung und Produktion moderner Erzeugnisse der elektronischen Nachrichtentechnik für den Zeitraum bis 1995 ab. Dabei geht es um einen höheren Beitrag sowie um die effektive Nutzung der Entwicklungs- und Produktionskapazitäten aller RGW-Länder, damit den Volkswirtschaften entsprechend ihren wachsenden Erfordernissen mehr elektronische Technik bereitgestellt werden kann. *ADN*

Erster gemeinsamer Betrieb der UdSSR und der DDR

Ein Abkommen zur Gründung eines gemeinsamen Betriebes der DDR und der UdSSR wurde während der Leipziger Frühjahrsmesse vom Minister für Elektrotechnik und Elektronik der DDR, Felix Meier, und dem sowjetischen Minister für Gerätebau, Automatisierungsmittel und Steuerungssysteme, M. Schkabardnja, geschlossen. Das gemeinsame deutsch-sowjetische Unternehmen soll ein *Wissenschaftliches Produktionszentrum zur Entwicklung, Lieferung und Wartung von Software* sein. Auch Datenverarbeitungssysteme werden entwickelt. Der Betrieb, der Anfang Mai seine Arbeit aufnahm, heißt *Zentron* und hat seinen Sitz in Kalinin.

Zentron ist ein Gemeinschaftsunternehmen des Kombinates Robotron und der wissenschaftlich-technischen Vereinigung Zentroprogrammsistem. Von seiten Robotrons wird der Betrieb mit Personalcomputern, 32-Bit-Rechnern sowie peripherer Schreib- und Vervielfältigungstechnik ausgerüstet. Auch ingenieurtechnisches Personal wird dabeisein. Die sowjetische Seite stellt die Räume und den größten Teil der etwa 200 Beschäftigten.

Das Unternehmen wird programmtechnische Komplexe und Programmsysteme für verschiedene Bereiche der Volkswirtschaft herstellen, für die ein großer Bedarf besteht. Zunächst werden Bestellungen der UdSSR und der DDR berücksichtigt, später auch die von Drittländern. Noch in diesem Jahr sollen die Anwender etwa 250 Systeme schlüsselfertig erhalten. Zentron arbeitet vollständig auf der Basis wirtschaftlicher Rechnungsführung. Das Grundkapital soll fünf Millionen transferable Rubel ausmachen. Jeder bringt die Hälfte ein. Das leitende Hauptorgan wird ein Rat sein, dem drei Vertreter aus jedem Land angehören. Dieser gemeinsame Betrieb ist einer der ersten in der UdSSR. Erst seit jüngster Zeit werden sie auch in der UdSSR gegründet. 1987 beschloß der Ministerrat der UdSSR den Modus der Bildung und Tätigkeit gemeinsamer Betriebe der UdSSR und anderer Länder auf ihrem Territorium. Seine Bestimmungen tragen den Interessen der Partner Rechnung und sehen beiderseitigen Vorteil vor. Die Gemeinschaftsunternehmen erarbeiten und bestätigen die Pläne für ihre Wirtschaftstätigkeit selbständig. Sie bestimmen selbst, was entwickelt und welche Erzeugnisse hergestellt werden sollen. Der neue Betrieb soll der wachsenden Bedeutung der Softwareproduktion bei der Anwendung der modernen Datenverarbeitung Rechnung tragen und den Kunden hochwertige Systemleistungen sowie einen hochwertigen Kundendienst anbieten. *APN/ADN*

Leistungswachstum in der Elektrotechnik/ Elektronik

Im Bereich der Elektrotechnik/Elektronik der DDR wurden im ersten Quartal Erzeugnisse für 156,4 Millionen Mark über den Plan hinaus gefertigt. Dieses Ergebnis wurde im April bei einem Erfahrungsaustausch der General- und Betriebsdirektoren der 15 Kombinate des Bereichs sowie der Direktoren der Außenhandelsfirmen in Berlin bilanziert. Jeder der für die volkswirtschaftliche Dynamik wichtigen etwa 240 Betriebe der Elektrotechnik/Elektronik sollte mit guter Planerfüllung im 40. Jahr des Bestehens der DDR Kurs auf den XII. Parteitag der SED nehmen, unterstrich der Minister für Elektrotechnik und Elektronik, Felix Meier. Überall komme es darauf an, noch stärker Wissenschaft, Technik und Rationalisierung einzusetzen und mehr Finalerzeugnisse ökonomisch zu fertigen. Das gelte auch für die Herstellung moderner, von der Bevölkerung gefragter Konsumgüter. Eingeschlossen sei dabei stets die Verbesserung der Arbeits- und Lebensbedingungen für die rund 490000 Werktätigen im Industriebereich. Durch kontinuierliche Planerfüllung hätten sich zum Beispiel die Kombinate VEB Carl Zeiss JENA, Automatisierungsanlagenbau, Mikroelektronik und Elektronische Bauelemente ausgezeichnet. Felix Meier hob hervor, daß besonders in der Technologie weitere Fortschritte erreicht werden müssen. Gute Beispiele dafür seien die Schaffung von Grundlagen zur Produktion schneller 16-Bit-Mikroprozessorsysteme sowie die Musterproduktion von 1-Megabit-Speichern. Der Minister verwies darauf, daß für die Mikroelektronik weitere neue Basistechnologien für kleinere Strukturabmessungen zu schaffen sind. Dafür sei der Aufbau eines leistungsfähigen Elektronikmaschinenbaus als Gebot der Entwicklung von Schlüsseltechnologien und als entscheidende Grundlage für künftige Investitionen unentbehrlich. Das habe die Produktion fester Widerstände im Kombinat Elektronische Bauelemente Teltow gezeigt. 60 bis 70 Prozent aller Bauelemente auf Leiterplatten im Industriebereich sind feste Widerstände. In Teltow sei es zudem gelungen, bei etwa 43000 verschiedenen Erzeugnissen für 4500 Abnehmer in der DDR die durchgängige rechnergestützte Bilanzierung, Auftragsannahme und Produktionssteuerung durchzusetzen. Vergingen früher bei 150000 Verträgen etwa sechs Monate von der Auftragsannahme bis zur -bestätigung, so seien es heute – dank Rechentechnik aus der DDR – zehn Tage. *ADN*

Ungarischer Handelsminister würdigt Wirtschaftsbeziehungen mit der DDR

Als nützlich und inhaltsreich hat der ungarische Handelsminister Tamas Beck die jüngste Tagung des Gemeinsamen Wirtschaftsausschusses DDR–Ungarn und die dabei getroffenen Vereinbarungen charakterisiert. Gegenüber der Zeitung „Magyar Hirlap“ verwies er in einem Interview darauf, daß der Warenaustausch zwischen Ungarn und seinem drittgrößten Handelspartner 1988 erstmals einen Wertumfang von mehr als zwei Milliarden Rubel erreichte. Beck sprach sich für den Ausbau der Direktbeziehungen zwischen Kombinaten, Betrieben und Institutionen beider Länder aus. Als positives Beispiel erwähnte er unter anderem die Zusammenarbeit zwischen der ungarischen Firma Videoton und dem Kombinat Robotron. *ADN*

Kopieren leicht gemacht

Ein Abkommen zwischen der sowjetischen Verlagsorganisation und Rank Xerox Ltd. bildet die Grundlage zur Eröffnung von sogenannten Copy-Shops in der Sowjetunion. Rank Xerox wird die Dienstleistungseinrichtungen ausrüsten und das Personal schulen. Die zunächst in Moskau eröffneten „Repro-Centers“ können von jedermann – Ausländer gegen konvertierbare Währung, sowjetische Bürger und Institutionen gegen Rubel – genutzt werden. *MP*

Robotron eröffnete Technisches Zentrum in Warschau

Das erste Technische Zentrum des Kombinates Robotron in der Volksrepublik Polen ist im April in Warschau eröffnet worden. Wie der stellvertretende Kombinatsdirektor Joachim Adrian bei der Eröffnung erklärte, solle es die Öffentlichkeitsarbeit auf dem polnischen Markt intensivieren und zugleich die Zusammenarbeit mit den Anwendern von Robotron-Technik vertiefen. Dazu ist das Zentrum mit den jeweils neuesten Erzeugnissen der Computer-, Schreib- und Meßtechnik des Unternehmens ausgerüstet. Von dem deutsch-polnischen Kollektiv aus Anwendungstechnikern und Kundendienstingenieuren wurde bereits ein Kursus zur Betriebsdatenerfassung veranstaltet. Neben diesen Aufgaben wird das Technische Zentrum auch Service- und Reparaturleistungen übernehmen und die 200 polnischen Firmen, die bereits einen Robotron-Kundendienst bieten, unterstützen. *ADN*

Suhler Softwarebörse

Rund 130 nachnutzbare Programme für Klein- und Personalcomputer wurden im März auf der zweiten Suhler Softwarebörse vorgestellt. Ausrichter der Veranstaltung in der Stadthalle der Freundschaft waren das Bezirksneuererzentrum, die Bezirksplankommission, die Kammer der Technik sowie das Kultur- und Sportzentrum der Bezirksstadt. Über 25 Betriebe und Einrichtungen des Bezirkes Suhl, darunter das Fahrzeug- und Jagdwaffenwerk Suhl und Robotron-Elektronik Zella-Mehlis stellten dabei interessante Softwarelösungen vor, die vorwiegend Arbeiten in der Betriebswirtschaft – wie Bilanzierung, Grundmittelrechnung und Arbeitskräftestatistik – wesentlich erleichtern. Das Südthüringer Fleischkombinat beispielsweise richtet gegenwärtig eine gemeinsam mit dem Datenverarbeitungszentrum entwickelte Datenbank ein, die künftig von allen fleischverarbeitenden Betrieben der DDR genutzt werden kann. *ADN*

Steigerung bei Japans Elektronikproduktion

Japans Elektronikproduktion hat 1988 gegenüber dem Vorjahr wertmäßig um 13 Prozent zugenommen. Wie die Assoziation der Elektronikindustrie (EIAJ) im März in Tokio bekanntgab, betrug sie am Jahresende 21,19 Billionen YEN (umgerechnet fast 303 Milliarden Mark). Über dem Durchschnitt lagen die Zuwachsraten bei der Fertigung von Computern (15,4 Prozent) sowie von Halbleitern und elektronischen Industrieausrüstungen (jeweils 14,4 Prozent).

Im Jahresbericht wird darauf verwiesen, daß der Import vor allem von Farbfernsehgeräten, Videorecordern und Stereoanlagen aus Ländern Südostasiens mit 57,9 Prozent ein Rekordwachstum erreichte. Aucch die Einfuhren elektronischer Komponenten, besonders von Schaltkreisen, nahmen mit 21,6 Prozent deutlich zu. Diese Entwicklung wird auf weitere Produktionsverlagerungen japanischer Konzerne ins Ausland und die spürbar gewachsene Kapazität in Südkorea, Singapur, Taiwan und Hongkong zurückgeführt. *ADN*

Aus 2 mach 3!

Aufmerksamen Lesern wird nicht entgangen sein, daß es in der Meldung „Intel 80486 angekündigt“ in MP 6/89, S. 188, beim Lieferstart des Prozessors mit 33 MHz Taktfrequenz nicht um den 80286-Prozessor, sondern um den 80386 ging. Wir bitten den Druckfehler zu entschuldigen.

Redaktion MP

Logischer Entwurf von Datenbanken

Thomas Zeier, Cottbus

Thomas Zeier (22) besuchte von 1983–1985 die EOS und arbeitete danach während eines Vorpraktikums im Rechenzentrum des Textilkombinates Cottbus als Operator und Programmierer. Er studiert seit 1987 an der TU Dresden Verarbeitungs- und Verfahrenstechnik und beschäftigt sich gegenwärtig mit rechnergestützten Untersuchungen der Oberflächenstrukturen von Fließgeweben.

Datenbank-Management-Systeme haben sich im Bereich der Mikrocomputer als besonders leistungsfähige Software-Werkzeuge erwiesen. Aufgrund der großen Flexibilität solcher Systeme kann, ein guter Entwurf vorausgesetzt, auf wechselnde Informationsbedürfnisse der Benutzer mit wenigen und geringfügigen Änderungen der Software reagiert werden.
Im Gegensatz zu einfachen Dateiverwaltungen besteht eine Datenbank nicht nur aus den einzelnen Daten, sondern sie enthält zusätzlich auch logische Verbindungen zwischen den Daten. Damit stellt sich eine Datenbank als Gesamtheit der vom Programm verwalteten Daten und deren Beziehungen zueinander dar. Diese Beziehungen können sehr komplex sein. Trotzdem müssen sie in der Struktur der Datenbank voll zum Ausdruck kommen.
Es wird gezeigt, wie Datenelemente so zu Datengruppen zusammengefaßt werden können, daß die entstehenden Datenstrukturen stabil und unanfällig gegenüber Änderungen sind. Auf der Grundlage derartiger Datenstrukturen wird einerseits ein Höchstmaß an Flexibilität bei der Verwaltung der Daten geboten. Andererseits halten sich der Aufwand für die Pflege von Anwendersoftware sowie das Ausmaß von Softwareänderungen in vernünftigen Grenzen. Die, mit Ausnahme der Schlüssel, redundanzfreie Abspeicherung aller Datenelemente unterstützt eine effiziente Auslastung des Speicherplatzes sowie möglichst kurze Zugriffszeiten.
Der Prozeß der Zusammenfassung von Datenelementen zu Datengruppen läßt sich durch die Überführung der Datenstruktur nacheinander in 3 abgeleitete Strukturformen gliedern /1/.

Die Datenbank in ihrer ursprünglichen Form

Als Beispiel möchte ich ein Schlagwortregister für verschiedene Zeitschriften wählen. Prinzipiell aber kann das Verfahren auch auf alle anderen Strukturen einer Datenbank angewendet werden. Ganz einfach stellen wir uns solch ein Register als eine Tabelle gemäß Bild 1 vor.

Titel	Autor 1	Autor 2	Autor 3	Schlagwort 1	Schlagwort 2	Schlagwort 3	Heftname	Nummer	Jahrgang
Anwendungsbsp.des ...	Müller	Meier		Taktgeber	B 555		Funkamateur	7	1988
Taktgeber mit...	Lehmann			D 100	Taktgeber		Funkamateur	6	1988
Kontrolleuchte...	Meier			LED			Originalbauplan	60	

Bild 1 Das Schlagwortregister in Tabellenform

In dieser Tabelle entsprechen die Spalten den Datenfeldern und die Zeilen den Datensätzen. Die erste Zeile stellt die Namen der Datenfelder dar. Über diese Namen können die einzelnen Datenfelder angesprochen werden. Die übrigen Zeilen sind der eigentliche Inhalt der Datenbank. Bild 2 ist das dazugehörige Schema. Der Name dieser Datei ist DATEN. Für jeden Artikel aus einer Zeitschrift wird wegen der Eindeutigkeit genau ein Datensatz verwendet. Die Nachteile dieser Struktur werden schon deutlich:

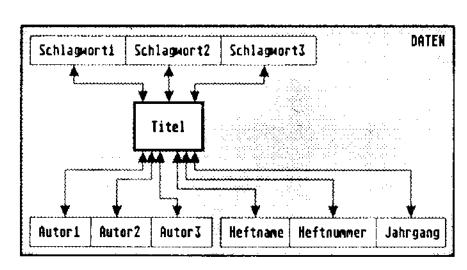


Bild 2 Die Datenbank in der ursprünglichen Form

- Eine begrenzte Anzahl von Autoren und Schlagwörtern pro Artikel stehen zur Verfügung. Was ist, wenn mehr Schlagwörter für einen Artikel benötigt werden, als vorgesehen sind?
- Wird nach einem bestimmten Schlagwort gesucht, dann muß zuerst im Datenfeld Schlagwort 1, dann in Schlagwort 2 und schließlich auch in Schlagwort 3 gesucht werden. Abgesehen von dem Programmieraufwand benötigt eine derartige Suche sehr viel Zeit. Analoges gilt auch für die Suche in den Datenfeldern Autor 1, Autor 2 und Autor 3.
- Gibt es weniger als 3 Autoren für einen Artikel, oder werden weniger Schlagwörter für einen Artikel benutzt, als vorgesehen sind, dann bleibt eine entsprechende Anzahl von Datenfeldern in diesem Datensatz leer. Tritt das oft auf, dann ist die Speicherplatzverschwendung enorm.

Diese Nachteile sollen genügen, um zu zeigen, daß die auf den ersten Blick recht gute Struktur doch zum Teil erhebliche Mängel aufweist. Wie kann man ihnen begegnen?

Die erste Strukturform

Zuerst wird eine laufende Nummer eingeführt. Das heißt, die Datensätze werden durchnumeriert. Der Anwender merkt später davon nichts. Sie dient nur der Auffindung der Daten, die zu einem Datensatz gehören. In anderen Datenstrukturen kann die laufende Nummer durch ein anderes Datenfeld, das den Datensatz eindeutig identifiziert, ersetzt werden. Dies kann z. B. die Nummer der Betriebe eines Kombinates sein.
Danach werden die Datenfelder, die inhaltlich übereinstimmen, zusammengefaßt und aus der eigentlichen Datei ausgelagert (Bild 3).
So entstehen die eigenständigen Dateien AUTOR und SCHLAGWORT. Die Doppelpfeile an ihren Datenfeldern Laufende Nummer zeigen, daß zu jedem Datensatz aus der Datei „DATEN" mehrere Datensätze der Dateien AUTOR bzw. SCHLAGWORT gehören können. Dies ist typisch für eine erfolgte Zusammenfassung von inhaltlich gleichen Datenfeldern. Die Datei AUTOR sieht also wie folgt aus:

Laufende Nummer	**Autorname**
1	**Müller**
1	**Meier**
2	**Lehmann**
3	**Müller**

Jetzt kann jeder Artikel beliebig viele Schlagwörter bzw. Autoren haben. Auch wenn nur ein Schlagwort oder Autor genügt, wird kein Speicherplatz verschwendet. Wird ein bestimmtes Schlagwort gesucht, dann braucht man nur einmal die Datei SCHLAGWORT zu durchsuchen. Man kann die gefundenen Datensätze zählen und erhält damit die Anzahl der Artikel, die dieses Schlagwort benutzen. Außerdem kann man über die entsprechenden den Einträge im Datenfeld Laufende Nummer auf Titel, Quelle und Autoren zugreifen. Analoges gilt auch, wenn ein bestimmter Autor gesucht wird. Diese Struktur ist also funktionstüchtig. Sie bietet sogar ein wenig Service. Zum Beispiel läßt sich mit **COUNT FOR <Bedingung>** sehr schnell die Anzahl der Artikel ermitteln, die ein bestimmtes Schlagwort benutzen, oder die ein bestimmter Autor geschrieben hat.

Die zweite Strukturform

Betrachtet man sich die Datei DATEN etwas näher, wird man feststellen, daß die Datenfelder Heftnummer und Jahrgang nur indirekt vom Titel abhängen. Ein Beispiel zeigt dies deutlich: Was sagt mehr über die Quelle aus: Anwendungsbeispiele des ...; 7; 88 oder Funkamateur; 7; 88?
Allgemein kann man sagen: Hängen Datenfelder indirekt vom Hauptschlüssel und direkt vom Nebenschlüssel ab, so werden sie zusammen mit dem Nebenschlüssel ausgelagert. In unserem Fall ist der Hauptschlüssel das Datenfeld Titel und der Nebenschlüssel das Datenfeld Heftname. Wieder wird zur Auslagerung ein Feld mit einer laufenden Nummer Buchnummer eingeführt (Bild 4). Es entsteht die Datei BUCH. Jedes Heft (z. B. Funkamateur 7/88) erhält eine Buchnummer. Die Anzahl der Datensätze in der Datei BUCH ist also gleich der der erfaßten Hefte.

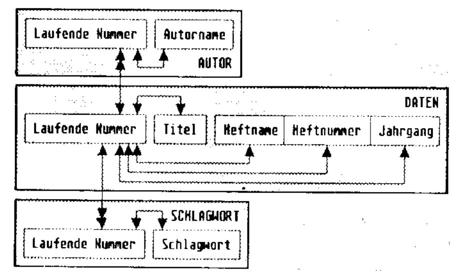


Bild 3 Die Datenbank in der ersten Strukturform

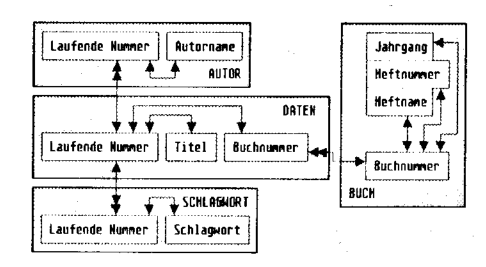


Bild 4 Die Datenbank in der zweiten Strukturform

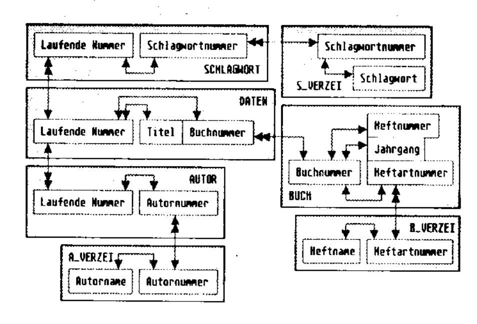


Bild 5 Die Datenbank in der dritten Strukturform

Die dritte Strukturform

In der Datei BUCH steht in unserem Beispiel unter Heftname zweimal Funkamateur und einmal Originalbauplan. In den Datenfeldern Autorname, Schlagwort und Heftname werden Daten mehrfach gespeichert. Dies stellt eine Datenredundanz dar. Sie kann beseitigt werden, denn sie benötigt zu viel Speicherplatz und eine eventuelle Änderung dieser redundanten Daten muß in der gesamten Datei mehrfach durchgeführt werden. Dieser Redundanz kann man mit einer Zusammenfassung der Daten eines Datenfeldes und deren Auslagerung begegnen. Die entstehende Struktur ist in Bild 5 dargestellt. Die entstehenden Dateien S_VERZEI, A_VERZEI und B_VERZEI stellen Verzeichnisse dar. Zum Beispiel ist in der Datei B_VERZEI jeder Heftname mit seiner zugehörigen Nummer genau einmal vertreten. Die Inhalte des Datenfeldes Heftartnummer der Datei BUCH stellen also Zeiger dar, die auf Datensätze des Verzeichnisses B_VERZEI zeigen. Die auf diese Weise entstandene Struktur ist um ein mehrfaches komplizierter als die Ausgangsstruktur. Sie zeichnet sich aber durch effektive Speicherplatzausnutzung und ständige Erweiterbarkeit aus.

Literatur

/1/ Langbein, D.; Eckardt, W.; Manecke, H.: Zur Theorie und Praxis der Anwendung des Datenbanksystems dBASE II. Kammer der Technik Suhl, TH Ilmenau 1987

Floppy-Treiber mit FDC U 8272 im Nicht-DMA-Modus Teil 2

Dr. Christian Löber
VEB Robotron Meßelektronik „Otto Schön" Dresden

FD-Treiber im ND-Poll-Modus
Grundforderungen an den Algorithmus

– schnelles, sicheres Positionieren, das heißt Eliminieren der Einflüsse durch vertikales (beim Kopfaufsetzen) bzw. horizontales (nach STEP-Operationen) Kopfschwingen
– zeitoptimierte Diskettenzugriffe
– Fehlertoleranz gegenüber Disketten- und Laufwerkstoleranzen (begrenzte Wiederholmechanismen mit gegebenenfalls Neupositionierung)
– Verifikation der Sektordaten (optionales Kontrollesen nach Schreiben)
– integrierte Überwachungsfunktionen (Vermeidung von Endlosschleifen bei Verklemmungen, undefinierten Systemzuständen, Fehlbedienungen, Diskettenfehlern)
– einfache, übersichtliche Struktur zur Erleichterung der Fehlersuche (manche Diskettentreiber sind sehr komplex und unstrukturiert programmiert).

Rufauswahl

Von den 15 FDC-Befehlen werden bei einem auf dem ND-Poll-Prinzip beruhenden FD-Controller für die Grundoperation Programmieren des FDC, Kopfpositionieren sowie Schreiben/Lesen Sektordaten nur acht benötigt.

① **SPEZIFY** (Code 03H)
Mit diesem Befehl werden die Betriebsart (DMA- bzw. Nicht-DMA-Modus) und die Timer für die Kopfladezeit HLT, Kopfentladezeit HUT und Schrittzeit SRT des FDC programmiert. Dieser Befehl sollte vor jeder Diskettenoperation gegeben werden. Die programmierten Werte werden durch FDC-Reset oder Kaltstart nicht gelöscht. In Bild 7 sind die für den Laufwerkstyp K 5601 optimierten Parameter angegeben. Sie gelten für Disketten im Format 5 × 1024 mit physischem Sektorversatz 1 4 2 5 3. Da beim Typ K 5601 der Kopf ständig aufgesetzt ist, sobald das Laufwerk geschlossen wird, könnte vermutet werden, daß der optimale Wert für HLT Null ist. Da die FDC-Zähler jedoch als Rückwärtszähler fungieren, führt HLT = 0 zum Maximalwert 508 ms (und damit insbesondere zu irritierend hohen Zeiten bei der Überwachung der Laufwerksaktivitäten!). Deshalb ist HLT = 01 der optimale Wert ensprechend einer Kopfladezeit von 4 ms bei der 4-MHz-Version U 8272 D04.

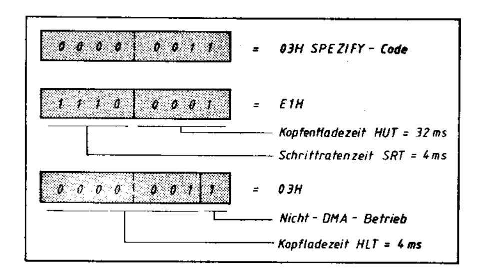


Bild 7 Parameterwerte für das SPEZIFY-Kommando für Laufwerkstyp K 5601 und 4-MHz-FDC U 8272 D04

② **RECALIBRATE** (Code 07H)
Beim Erstzugriff auf ein Laufwerk und in manchen Fehlersituationen (mechanisch fehlpositionierter Kopf) muß der interne Spurzähler PCN des FDC mit der tatsächlichen Kopfposition in Übereinstimmung gebracht werden. Das geschieht durch den Befehl RECALIBRATE, wobei die Spur 0 angefahren wird. Da der FDC in einem Zug nur maximal 77 Schritte machen kann, muß bei 80spurigen Disketten das Anfahren der Spur 0 in zwei Etappen vollzogen werden. Das Erreichen der Spur 0 kann durch Abfrage des Laufwerkstatus überprüft werden (B4/ST3 = 1).

③ **SENSE DRIVE STATUS** (Code 04H)
Vor dem Erteilen eines Positionierungskommandos ist mit diesem Befehl die Bereitschaft des Laufwerkes zu testen (B5/ST3 = 1). Ein Laufwerk ist bereit, wenn der Laufwerkshebel geschlossen, der Motor Sollgeschwindigkeit hat und die Diskette richtig gesteckt ist.

④ **SEEK** (Code 0FH)
Neben Drive- und Kopfnummer ist beim Suchkommando in einem weiteren Byte die anzufahrende Spurnummer NCN an den FDC zu übergeben. Der FDC erzeugt dann so viele Schrittimpulse, bis Istspur PCN = Sollspur NCN. Im allgemeinen wird empfohlen, nach einem SEEK über das Kommando READ ID ein ID-Feld der angefahrenen Spur zu lesen, wobei neben ST0 ... ST2 die vollständige Diskettenadresse des Sektors (Spur, Kopf, Sektornummer, Sektorgröße) als Resultat übergeben wird und somit die tatsächlich erreichte Spur kontrolliert werden kann. Das führt jedoch zu Zeitverlusten, deshalb sollte darauf verzichtet werden, da bei nachfolgenden Lese- oder Schreibbefehlen das Identifikationsfeld (ID-Feld) ohnehin noch einmal gelesen und verglichen wird.

⑤ **SENSE INTERRUPT STATUS** (Code 08H)
Von den zahlreichen zur Generierung eines Interrupts führenden Ereignissen sind hier wegen des interruptfreien Grundprinzips nur die Interrupts infolge der Kommandos 3 und 4 – beide haben keine Ergebnisphase – wichtig. Jedem dieser Befehle muß Kommando 5 folgen, um über die zurückgemeldeten Informationen ST0 und PCN das Ergebnis überprüfen zu können und um das Interrupt-(IR-)-Signal zurückzusetzen.

Dabei gilt folgende Zuordnung für ST0:

IR-Code		Suchende-bit	
B7	B6	B5	
0	0	1	fehlerfrei beendetes Kommando
0	1	1	mit Fehler beendetes Suchkommando (z. B. Spur nicht gefunden)
1	1	0	RDY-Wechsel (z. B. Laufwerkshebel geöffnet)

MT MFM SK 0 0 1 1 0	46H	Befehlskode für Lesen
0 0 0 0 0 HD DS1 DS0	05H	Kopf 1, Laufwerk 1
C	10H	Spurnummer (Zylinder)
H	01H	Kopfnummer (Seite)
R	01H	Sektornummer
N	03H	relative Sektorlänge 2^N, N = 0 ... 3
EOT	05H	Nummer des letzten Sektors der Spur
GPL	38H	Anzahl der Lückenbytes in GAP 3
DTL	FFH	Datenfeldlänge (bei N > 0 = FFH)

Bild 8 Befehlssatz für das Lesen von 1-KByte-Sektoren der Spur 16 ab Sektor 1
(MT = 0 kein Spurverlängerungsverfahren, d. h. Kopfverwaltung im Blockungsmodul;
GAP3 = 38H empfohlene Lückenlänge bei 1-KB-Sektoren;
SK = 0: Arbeit nur mit Sektoren mit Datenmarke FBH;
DTL für Sektoren > 128 Byte ohne Bedeutung)

⑥ **READ DATA** (Code 46H: MFM-Betrieb, keine Spurverlängerung)
Leseoperationen werden durch einen Satz von 9 Parametern spezifiziert – ein Beispiel für eine Sektorleseoperation zeigt Bild 8. Zu Beginn der Ausführungsphase wird der Kopf aufgesetzt, entsprechend der im SPEZIFY-Befehl programmierten Kopfladezeit HLT gewartet und nach Erkennen des Indexloches mit dem Lesen und Vergleichen der ID-Informationen der angefahrenen Spur begonnen. Nach gefundenem ID-Feld liest der FDC die Daten des Sektors. Der Prozessor muß spätestens 24 μs nach Eintritt der Datenbereitschaft (RQM-Bit7/FDCSTAT = 1) reagieren und Byte für Byte im 32-μs-Rhythmus übernehmen, ansonsten wird das Overrun-Flag B4/ST1 gesetzt und die Operation vom FDC beendet. Das normale Ende einer Sektoroperation ist die Ausgabe eines TC-Impulses per Programm.
Im ND-Modus setzt der FDC nach Empfang jedes Datenbytes das INT-PIN sowie das RQM-Bit des Hauptstatusregisters. Beide werden durch das Auslesen des empfangenen Bytes durch den Prozessor zurückgesetzt. Das eröffnet die Möglichkeit, im reinen Polling arbeiten zu können, das heißt, das Bytehandling wird nur durch Abfrage des RQM-Bits gesteuert (ND-Poll-Prinzip), und auf eine komplizierte IR-Steuerlogik wird verzichtet. Zum Erkennen von möglichen „Verklemmungen" des FDC – das kann beispielsweise durch Öffnen des Laufwerkshebels provoziert werden – sollte die diesbezügliche Abfrageschleife für das RQM-Bit zeitüberwacht werden.
Beim 2,4-MHz-Prozessor UB 880 ist dies – wegen des bereits erwähnten 32-μs-Zyklus – nur interruptgesteuert möglich (z. B. mittels eines CTC-Kanals). Beim 4-MHz-Prozessor UA 880 kann die Überwachung dagegen aufgrund der höheren Verarbeitungsgeschwindigkeit rein softwaremäßig geschehen. Das hat den großen Vorteil, daß die gesamte Sektoroperation in eine DI/EI-Klammer eingeschlossen werden kann, um die Sektorarbeit vor Unterbrechung durch systeminterne oder -externe IR-Quellen zu schützen. Bei der 2,4-MHz-Variante hingegen müssen für die Dauer der Sektorarbeit alle (also auch später gegebenenfalls durch den Anwender nachdefinierte IR-Quellen!) für die Dauer der Sektorarbeit abgeschaltet werden.
⑦ **WRITE DATA** (Code 45H)
Analog zu READ DATA, jedoch vor Auslösen des Rufes Test, ob Diskette schreibgeschützt ist durch Ruf 3 (Abfrage B6/ST3).
⑧ **SCAN EQUAL** (Code 51H)
Dieser Befehl kann dazu benutzt werden, nach Sektorschreiboperationen die Read-After-Write-Funktion (RAW-Funktion = Kontrollesen) auszuführen. Jeder geschriebene Sektor sollte prinzipiell – das heißt ohne Berücksichtigung eventueller Kontrollmechanismen in den einzelnen Dienstprogrammen eines Betriebssystems – vom Treiber kontrollgelesen werden, trotz des dafür erforderlichen Zeitaufwandes, da die Datensicherheit Vorrang vor der Datenübertragungsgeschwindigkeit hat. Gezielt kann die RAW-Funktion außer Kraft gesetzt werden, indem B3/Byte 6 des im Requestvektor FD_REQ angegebenen Parameterblockes auf 1 gesetzt wird (siehe Bild 6 in MP 6/89, S. 168).
Bei SCAN EQUAL ist in der Ausführungsphase Byte für Byte aus dem RD/WR-Puffer an den FDC auszugeben. Dieser liest Byte für Byte des geschriebenen Sektors zurück und kontrolliert auf Gleichheit. Wenn alle Daten gleich sind, wird das Scan-Hit-Bit B3/ST2 gesetzt (wird in der Resultatsphase zurückgemeldet). Damit ist festgelegt, daß alle Daten einer vorangegangenen Sektorschreiboperation richtig abgespeichert wurden, mit einer Ausnahme: Ein fehlerhaft geschriebenes Byte FF wird nicht als Schreibfehler erkannt, da FF alle Vergleichsbedingungen erfüllt. Ein solcher Fehler wird jedoch über die CRC-Bytes erkannt.
⑨ **READ ID** (Code 4AH)
Bei Generierung dieses Rufes liest der FDC das als nächstes folgende ID-Feld auf der aktuellen Spur und liefert neben ST0, ST1 und ST2 Spur, Kopf, Sektor und Sektorlänge. Im allgemeinen wird der Befehl nach Positioniervorgängen zur Spurkontrolle eingesetzt. Das ist jedoch – um Zeit zu sparen – prinzipiell nicht nötig, da – wie schon erwähnt – bei nachfolgenden Schreib- oder Leseoperationen vom FDC vorher ohnehin das korrespondierende ID-Feld identifiziert werden muß. Deshalb wird READ ID für die reine Sektorarbeit nicht benötigt, wohl aber dann, wenn das zu generierende Betriebssystem automatisch das Diskettenformat erkennen soll. Dabei wird unter anderem im BIOS-Ruf SELDSK ein ID-Lesen zum Erkennen der tatsächlichen Sektorlänge einer gesteckten Diskette benutzt.

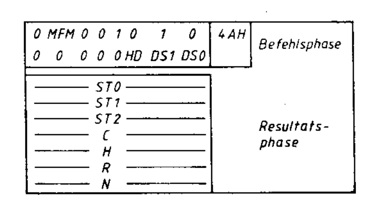


Bild 9 Befehlscode für ID-Feldlesen und zurückgegebene Parameter

Ausgewählte Algorithmen
Kommandoausgabe an den FDC

Nach **POWER ON**-Reset steht das Directionsbit DIO des Hauptstatusregisters auf Ausgabe (= Grundstellung), das heißt, der FDC ist für den Empfang des allerersten Kommandos – zum Beispiel vom Befehl SPEZIFY – bereit. Danach müssen alle Phasen eines ausgegebenen Kommandos: Befehls-, Ausführungs- und Resultatsphase durchprogrammiert werden. Insbesondere müssen in der Ergebnisphase alle Resultatbytes abgeholt werden (Kontrolle über Bit 4/Hauptstatusregister), ansonsten ist der FDC nicht für den Empfang weiterer Befehle bereit: Er bleibt in Richtung FDC ⇒ CPU stehen.

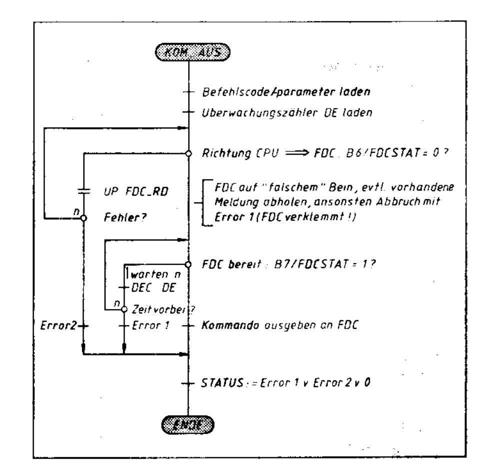


Bild 10 PAP für die Ausgabe eines Kommandos mit Test, ob FDC ggf. auf Abholen einer Meldung durch den Prozessor wartet

Ebenso schaltet der FDC auf das Senden einer Message um, wenn im Ergebnis der laufenden Bereitabfrage der Laufwerke eine Zustandsänderung auftritt (z. B. RDY-Wechsel). Beide Situationen müssen in einem Programm zur Kommandoausgabe an den FDC berücksichtigt sein (Bild 10 und 11). Des weiteren sollten die Programme zum Ausgeben eines Kommandos und zum Lesen des FDC (Daten oder Resultat einer Operation) zeitüberwacht werden, um Endlosabfrageschleifen bei undefinierten Systemzuständen mit einer definierten Fehlerbehandlung abzubrechen.

Positionieren

Beim Spursuchen (Bild 12) sind einige Besonderheiten zu beachten:

– Vor jedem Suchbefehl ist die Laufwerksbereitschaft mit dem Kommando **SENSE DRIVE STATUS** (B5/St3 = 1: Laufwerk bereit) zu testen.

– Bei Kaltstart sowie beim Neupositionieren infolge fehlerhafter Lese- oder Schreiboperationen sind alle Variablen bzw. ist die betreffende Variable für die logischen Istspuren der Laufwerke auf eine physisch nicht mögliche Spurnummer – beispielsweise FFH – zu initialisieren, um beim Erstzugriff oder bei bestimmten Rufwiederholungen für das betreffende Laufwerk ein HOME über das Kommando RECALIBRATE und damit das Kalibrieren des internen Spurzählers PCN des FDC auf Spur 0 zu veranlassen. Dabei ist zu beachten, daß im Unterschied zum SEEK-Befehl der RECALIBRATE-Ruf eine maximale Spurdistanz von nur 77 Schritten überbrücken kann. Deshalb muß Programm HOME bei zu überbrückenden Spurdistanzen > 77 Spuren eine Doppelpositionierung durchführen.

– Nach einem **SEEK**-Befehl ist eine Wartezeit von 5–10 ms zum Ausgleich der horizontalen Kopfschwingungen am Ende des Positioniervorganges abzuwarten. Danach ist mit einem SENSE IR-STATUS-Kommando das Ergebnis ST0 und PCN des SEEK-Befehls abzuholen. Dabei werden gleichzeitig das entsprechende Drive-Status-Bit D0 ... D3 des Hauptstatusregisters und die INT-Leitung zurückgesetzt.

– Das FDC-BUSY-Bit 4 des Hauptstatusregisters wird – im Unterschied zu Lese- und Schreibbefehlen – bei Suchbefehlen schon am Ende der Befehlsphase zurückgesetzt; es ist folglich für das Erkennen des Endes der Ausführungsphase nicht geeignet.

– Wird bei einem Suchbefehl durch Öffnen des Laufwerkhebels die RDY-Leitung auf *NICHT BEREIT* gesetzt, so unterbricht der FDC den Positioniervorgang. Dabei auftretende undefinierte Systemzustände können durch Wiederholmechanismen mit dem Neupositionieren über HOME eliminiert werden.

Sektorarbeit

In Bild 13 ist der dem FD-Treiber zugrundeliegende Grundalgorithmus mit einem zweifachen Wiederholungsmechanismus bei auftretenden Fehlern dargestellt. Alle angeführten Unterprogramme enthalten eine eigene Zeitüberwachung nach dem Softloop-Prin-

```
                     ;################################
                     ;   KOMMANDO_AUSGABE AN FDC      #
                     ;################################
                     ;
0040                 FDCSTAT   equu   40h            ;Portadr.Hauptstatusregister FDC
0041                 FDCDAT    equ    FDCSTAT + 1    ;Portadr.Datenregister FDC
0001                 ERROR1    equ    01             ;FDC nicht bereit (Zeitueberschreitung)
0002                 ERROR2    equ    02             ;FDC verklemmt (weder Ausgabe eines Be-
                                                     ;fehls noch Lesen einer Meldung moeglich)
0000'   0E 41        KOM_AUS: ld      c,FDCDAT
0002'   3A 0036'              ld      a,(BEFZAL)     ;Anzahl Befehlsbytes
0005'   47                    ld      b,a
0006'   21 0037'              ld      hl,KOM_TAB     ;Befehlssatz
0009'   11 0000               ld      de,0           ;Ueberwachungszaehler
                     ;
000C'   DB 40        direc?: in       a,(FDCSTAT)
000E'   CB 77                bit      6,a            ;DIO-Bit: Richtung CPU ==> FDC ?
0010'   C4 0028'             call     nz,FDC_RD      ;nein:    Richtung CPU <== FDC !
0013'   20 12                jr       nz,ende
                     ;
0015'   DB 40        rdy?:   in       a,(FDCSTAT)
0017'   CB 7F                bit      7,a            ;RQM-Bit: FDC bereit (B7=1) ?
0019'   20 09                jr       nz,ausg        ;ja: zur Befehlsausgabe
                     ;
001B'   1B                   dec      de             ;nein: Ueberwachung dekrementieren
001C'   7A                   ld       a,d
001D'   B3                   or       e              ;Ueberwachungszeit vorbei?
001E'   20 F5                jr       nz,rdy?        ;nein: weiter abfragen
0020'   3E 01                ld       a,ERROR1       ;ja: Fehlerart 1 markieren u.zum Ende
0022'   18 03                jr       ende
0024'   ED A3        ausg:   outi                    ;Befehlssatz ausgeben
0026'   AF                   xor      a              ;STATUS: kein Fehler
0027'   C9           ende:   ret
                     ;
                     ;UP zum Lesen einer Nachricht vom FDC
                     ;-----------------------------------
0028'   DB 40        FDC_RD: in       a,(FDCSTAT)
002A'   E6 C0                and      0c0h           ;Maske
002C'   FE C0                cp       0c0h           ;Nachricht im FDC:   B7/B6_FDCSTAT = 11 ?
002E'   3E 02                ld       a,ERROR2       ;(Fehlerart 2 markieren)
0030'   C0                   ret      nz             ;nein: FDC-Verklemmung (schwerer Fehler!)
                     ;
0031'   CD 0035'             call     MESSAGE        ;ja: Meldung abholen, z.B.
                             ret                     ;    Statusaenderung auf RDY-Leitung
                     ;
0035'                MESSAGE: ; ...                  ;UP zur Abholung der Meldung (Dummy)
0035    C9                   ret
                     ;
                     ;Variable:
                     ;---------
0036'   09           BEFZAL:  db      9              ;Anzahl Bytes im Befehlssatz
0037'   03           KOM_TAB: db      46h            ;Befehlscode:  READ DATA
0038'   05                    db      05h            ;Kopf 1 / Laufwerk 1
0039'   10                    db      10h            ;Spur 16
003A'   01                    db      01h            ;Kopf 1
003B'   01                    db      01h            ;Sektor 1
003C'   03                    db      03h            ;1024 B/Sektor
003D'   05                    db      05h            ;Nr.letzter Sektor der Spur
003E'   38                    db      38h            ;GAP3
003F'   FF                    db      0ffh           ;'Datenfeldlaenge'
```

Bild 11 Programmbeispiel für Ausgabe eines Kommandos an den FDC

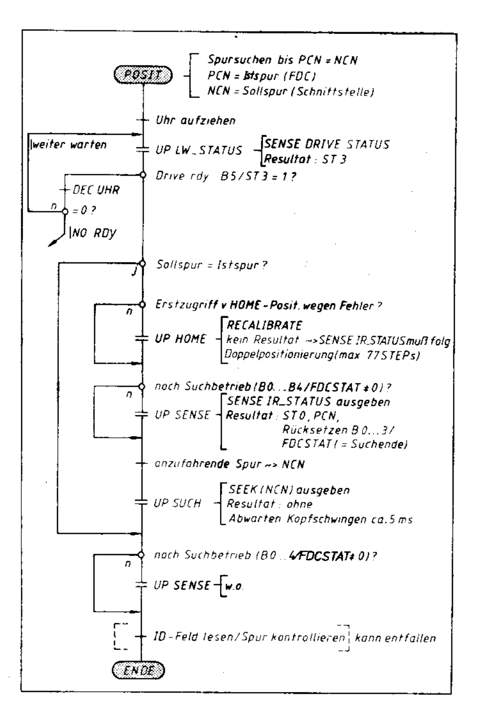


Bild 12 PAP für Kopfpositionierung über RECALIBRATE- bzw. SEEK-Befehl

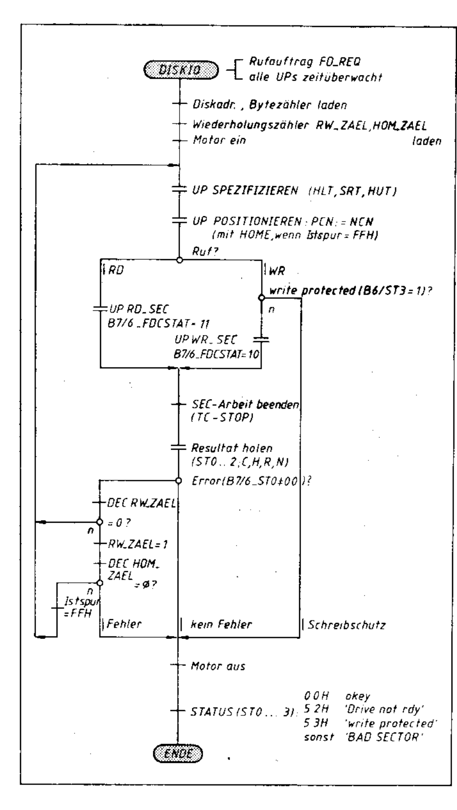


Bild 13 PAP für die DISK-I/O-Routine zum Lesen oder Schreiben von Sektordaten ▼

```
                     ;################################
                     ;   UP LESEN EINES SEKTORS       #
                     ;################################
                     ;
0040                 FDCSTAT   equ    40h
0041                 FDCDAT    equ    FDCSTAT + 1
0048                 ZREG      equ    48h            ;zusaetzliches Steuerregister
FFFF                 KOM_AUS   equ    0ffffh         ;s.Bild 11
                     ;
0000'   CD FFFF      SEC_RD: call     KOM_AUS        ;FDC auf Lesen Sektor programmieren
0003'   C2 0030'             jp       nz,rdend       ;zum Ende,wenn FDC nicht okey
0006'   ED 4B 0031'          ld       bc,(BYTZAL)
000A'   2A 0033'             ld       hl,(PUFF)
000D'   11 0000              ld       de,00          ;Ueberwachungszaehler laden
                     ;
                     ;BYTEHANDLING:
                     ;-------------
0010'   F3                   DI                      ;Oeffnen der DI/EI-Klammer
0011'   1B           rd:     dec      de
0012'   7B                   ld       a,e
0013'   B2                   or       d
0014'   28 19                jr       z,rd1          ;Austritt bei 'Zeitueberschreitung'
                     ;
0016'   DB 40        rd0:    in       a,(FDCSTAT)    ;Status des FDC holen
0018'   FE F0                cp       0f0h           ;B7B6B5B4 = 1111:      ==> Daten vorhanden
001A'   20 F5                jr       nz,rd          ;bei Desynchr.FDC/CPU:  ==> Rueckmeldung D0H
001C'   DB 41                in       a,(FDCDAT)     ;Byte lesen
001E'   77                   ld       (hl),a         ;und abspeichern
001F'   ED A1                cpi
0021'   EA 0016'             jp       pe,rd0         ;alle Daten gelesen ?
                     ;
0024'   3A 0433'             ld       a,(zwsp)       ;ja:   Beenden des Transfers
0027'   CB E7                set      4,a            ;durch Ausgabe TC-Stopimpuls
0029'   D3 48                out      (ZREG),a
002B'   CB A7                res      4,a
002D'   D3 48                out      (ZREG),a
                     ;
002F'   FB           rd1:    EI
0030'   C9           rdend:  ret
                     ;
0031'   0000         BYTZAL: dw       00             ;zu transferierende Bytezahl
0033'                PUFF:   ds       400h           ;Schreib/Lese-Puffer
0433'   00           zwsp:   db       0              ;Zwischenspeicher f.ZREG
                             END
```

Bild 14 Programmbeispiel für das Bytehandling beim Lesen aller Bytes eines Sektors

zip. Folgende Hauptaktivitäten charakterisieren den Treiber:

– Einschalten des aktuellen Motors (Berechnen des entsprechenden Motorcodes aus Byte 2 der übergebenen Schnittstelle FD_REQ) sowie Laden zweier Zähler für Rufwiederholung ohne bzw. mit Neupositionierung (RW_ZAEL, HOM_ZAEL).

– Innerhalb der Wiederholungsschleife prinzipiell Neuspezifizieren des FDC, das heißt, jeweiliges Neuladen von HLT, HUT und SRT und damit verbunden Verwendung bestimmter Rücksetzfunktionen des FDC; ferner Positionieren des Kopfes auf die in FD_REQ angewiesene Sollspur NCN; jedoch kein ID-Feldlesen zur Spurkontrolle aus Zeitgründen.

– Rufverteiler für das Trennen von RD- und WR-Rufen mit anschließendem Bytehandling. In Bild 14 ist hierfür ein konkretes Programmbeispiel für das Lesen eines Sektors angegeben. Die Ausführungsphase wird nach dem Übertragen aller Sektordaten (Test: Bytezähler = 0) durch die programmierte Ausgabe eines vollständigen > 4 μs langen High-Impulses an TC über das zusätzliche Steuerregister des FD-Controllers beendet.

– Lesen und Auswerten des Resultats der FDC-Operation. Bei aufgetretenen Fehlern (dann ist B7/B6_ST0 $\neq$ 00) erfolgt eine n-fache Rufwiederholung des Schreib-Lese-Rufes bis RW_ZAEL = 0, danach nochmalige (einmalige) Wiederholung mit Neupositionierung über Spur Null, um außergewöhnliche Fehlersituationen, wie undefinierte Stellung des Kopfes (z. B. auch außerhalb des Spurbereiches der Diskette!) zu beheben.

– Motorabschalten und Rückgabe des Status der Operation an den Blockungsmodul. Im Fehlerfall kann die konkrete Fehlerur-

sache durch Austesten der Statusregister des FDC detailliert ermittelt werden (z. B. B1/B4_ST2 = 1: schlechte oder falsche Spur, B5/ST1 = 1: CRC-Fehler, B0/ST1 = 1: fehlende Adreßmarke usw.). Bei den meisten CP/M-kompatiblen Betriebssystemen erfahren jedoch nur die Situationen *Drive not rdy* und *write protected* eine gesonderte, für den Bediener relevante Behandlung, während auf alle anderen Fehlersituationen überwiegend mit *Bad Sector* reagiert wird.

Meßergebnisse

Bei der Arbeit mit Floppy-Speichern sind für den Nutzer vor allem folgende drei Parameter bzw. Kriterien aussagekräftig:

● Die Zugriffssicherheit, das heißt die Häufigkeit von Fehlern, bei denen das Betriebssystem mit *Bad Sector* reagiert. Zur Ermittlung eines Orientierungswertes wurden Dauerkopierversuche zwischen zwei Laufwerken über jeweils 8 Stunden durchgeführt. Maßgeblichste Einflußgröße ist dabei die Diskettenqualität. Bei *einwandfreien* Disketten trat bei $\geqq 10^5$ Sektoroperationen kein Fehler auf.

● Die Zugriffsgeschwindigkeit, die direkt proportional in die Kopierzeiten eingeht. Hierbei wirken neben Maßnahmen in der Hardware (erhöhte Prozessortaktfrequenz, hardwaremäßige CRC-Sicherung durch den FDC entsprechend 7 ms Zeiteinsparung/ Sektorzugriff /2/ folgende softwareseitige Maßnahmen zugriffsbeschleunigend:

– kein ID-Feldlesen bei Positioniervorgängen

– optimierte Laufwerksparameter, insbesondere hinsichtlich der Schrittratenzeit SRT (4 ms + 1 ms zum Ausgleich von Laufwerkstoleranzen) und minimale Kopfladezeit HLT = 4 ms.

– große Sektorlänge (1-KByte-Sektoren, dies natürlich auch aus Gründen maximaler Diskettenkapazität) und Zugriffsverdichtung durch Blockung

– physischer Sektorversatz 1 4 2 5 3 beim Format 5 * 1024 Byte/Spur (realisiert im zugehörigen FORMAT-Programm). Im Zusammenwirken aller genannten Maßnahmen wurden die in Tafel 1 angegebenen Kopierzeiten erzielt. Zum Vergleich sind die analogen Werte für die Büro-und Personalcomputer der DDR angegeben.

● Die Fehlertoleranz, das heißt, in welchem Maße auch grobe Fehler toleriert bzw. unterdrückt werden. Derartige Einflüsse lassen sich zum Beispiel dadurch provozieren, daß während eines laufenden Kopiervorganges von Hand die Schlittenposition in starkem Maße verstellt wird, im kritischsten Falle sogar außerhalb des formatierten Spurbereichs der Diskette, wodurch der FDC weder eine Spur noch ein ID-Feld vorfindet. Unter Zugrundelegung der beschriebenen fehlertolerierenden Treiberphilosophie werden diese und andere Störungen problemlos ohne Fehlerausstieg eliminiert.

Rechner	BC A5120		PC 1715	A5105				EC 1834
Laufwerkstyp	K5600.10	K5601	K5601	K5601				K5601
Format	16 × 256		16 × 256	5 × 1024				9 × 512
physischer Sektorversatz (14253)				ohne	mit			
Kontrollesen				mit	mit	ohne	*)	
Kopierzeiten in s	124	117	105	102	81	61	60	60
s/KByte	1,60	1,50	1,35	1,31	1,04	0,78	0,77	0,77

Tafel 1 Kopierzeiten im Vergleich (Kopieren von 8 SCP-Dienstprogrammen = 78 KByte mit POWER unter vergleichbaren Bedingungen) *) Kopieren mit PIP

Literatur

/1/ Funktionsbeschreibung FDC U 8272. Entwurf der Schaltkreisdokumentation 30. 9. 1985

/2/ Böhl, E.: Integrierte Floppy-Disk-Controller-Schaltungen U 8272 D 08 und U 8272 D 04. Radio, Ferns., Elektron. 36 (1987) 11, S. 703

/3/ KROS 5110/01. Werkstandard VEB Kombinat Robotron, September 1985

Desktop Publishing

Jörg Simon

Die unaufhörliche und schnelle Entwicklung der Personalcomputer (PC) und der damit verbundenen Gerätetechnik in den letzten Jahren hat dazu geführt, daß neben die bisherigen generellen Nutzungsmöglichkeiten der PCs, wie Textverarbeitung, Tabellenverarbeitung, Datenbanken, Grafikverarbeitung oder Telekommunikation, eine neue, das Anfertigen von Publikationen mit PCs – Desktop Publishing (DTP) genannt – getreten ist. Der Artikel beschreibt die Voraussetzungen, die für eine solche Anwendung notwendig sind.

Zu den technischen Voraussetzungen für DTP zählen neben einem PC mit mindestens 640 KByte Hauptspeicher und einer Festplatte folgende Geräte:

● Scanner, mit dessen Hilfe Abbildungen (Fotografien, Texte, Grafiken) vom Papier in eine Datei, die von DTP-Software verarbeitet werden kann, kopiert werden

● Bildschirm mit hoher Auflösung, der möglichst eine ganze A4-Seite wahrheitsgetreu wiedergeben kann (Ganzseitenbildschirm)

● Maus, die sowohl zur Steuerung des DTP-Programms als auch zur Arbeit mit Grafik verwendet wird

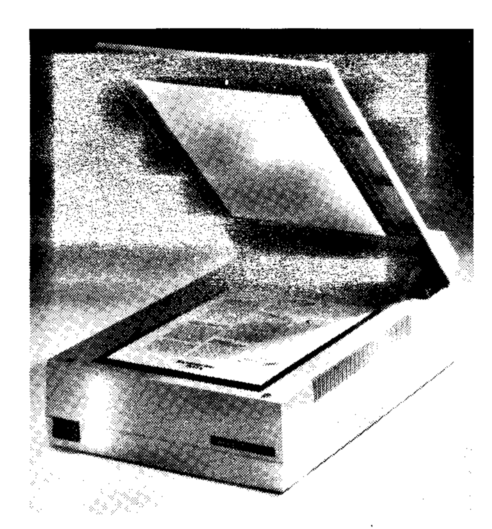
● Drucker von hoher Qualität, um eine ausgezeichnete Wiedergabe des Textes und der Abbildungen zu gewährleisten

● Software, die sowohl Text- als auch grafische Verarbeitung zuläßt, das gesamte Dokument typografisch aufbereiten kann und den Import von Dateien verschiedenen Formats in das Dokument ermöglicht.

In Abhängigkeit von den eigenen Ansprüchen und der vorhandenen Hard- und Software erlauben DTP-Systeme den Druck von Dokumenten von minimaler (Nadeldrucker mit neun Nadeln), mittlerer (Nadeldrucker mit 24 Nadeln) bis zu hoher Qualität (Laserdrukker mit hoher Auflösung). Entsprechend den Qualitätsanforderungen variieren auch die Preise von DTP-Systemen.

Scanner

Die überwiegende Anzahl von DTP-Anwendern startet, ohne einen Scanner zur Verfügung zu haben. Es dauert allerdings nicht lange, um zu erkennen, daß das Einfügen von Fotografien oder Grafiken in das eigene Dokument sehr wünschenswert wäre. Der Weg führt zum DTP-Scanner. Die Arbeitsweise eines Scanners beruht auf dem Abtasten einer Abbildung (Fotografie, Text, Grafik usw.) mittels eines lichtempfindlichen Geräts und der Konvertierung des resultierenden Signals in eine digitale Form. Scanner (z. B. Belegleser) sind seit langem als Eingabegeräte für Großrechner im Einsatz; allerdings erweisen sie sich als sehr kostspielig und sensibel. Der Einsatz in Verbindung mit Personalcomputern ist eine relativ junge Entwicklung. Waren 1986 in den USA ungefähr 4000 Scanner im Einsatz, so stieg diese Zahl 1988 auf etwa 36000. Das bedeutet, daß etwa einer von sieben DTP-Arbeitsplätzen mit einem Scanner ausgerüstet ist /1/. DTP-Scanner am unteren Ende der Preisskala liegen im Preisbereich von XT-kompatiblen PCs, während die an der Spitze rangierenden Scanner etwa zehnmal teurer sind. Der Unterschied liegt in der Auflösungsfähigkeit und der Fähigkeit,

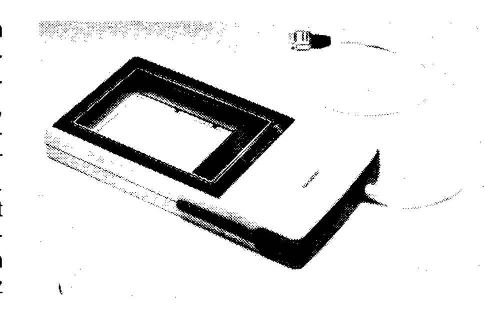


Mit einer Auflösung von 800 dpi (dots per inch – Punkte je Zoll) zählt der neue Flachbettscanner TDS 8000 von Tandberg Data zur gehobenen Leistungsklasse. Er tastet die Vorlagen mit einem CCD-Sensor ab; störanfällige Spiegel- und Linsensysteme wurden durch Faseroptik und LED-Beleuchtung ersetzt. Bei Umschaltung auf 400 dpi beträgt die Scanzeit 11 Sekunden. Die Scansoftware erlaubt das Einlesen von Text-, Bild- und Grafikvorlagen und das Mischen auf dem Bildschirm, außerdem Fenstertechnik für 16 unterschiedliche Scanparameter.

Texte unterschiedlicher Schriftsetzung in eine Textdatei konvertieren zu können (optische Zeichenerkennung). Die Auflösungsfähigkeit reicht von 200 bis 1000 Pixel per Zoll, in Verbindung mit der Breite der Grautonskala (2 bis 256 verschiedene Grautöne) für jeden Pixel ergibt sich die Qualitätsvarianz. Man bezahlt jedoch für die Bildqualität mit Speicherplatz. So belegt eine Abbildung (A4-Format), digitalisiert mit einer Auflösung von 300 Pixel per Zoll und 2 Grautönen (schwarz und weiß) mehr als 1 MByte, bei 256 verschiedenen Grautönen mehr als 8 MByte, bei einer Auflösung von 600 Pixel per Zoll und 256 Grautönen etwa 34 MByte Speicherplatz. Allerdings ist es auch eine Frage des verwendeten Druckers, ob die Eingabequalität in eine Druckausgabe hoher Qualität umgesetzt werden kann. Für die meisten Anwendungsfälle ist ein Scanner mit 64 Grautönen mehr als ausreichend. Interessant ist die Möglichkeit, die digitalisierten Abbildungen im PC manipulieren zu können. So kann der Grautonwert jedes Pixels geändert werden, Kontraste können verschärft und die Abbildungen retuschiert werden. Alle Scanner sind mit einer diesbezüglichen Software ausgerüstet. Ein anderes Softwarepaket, welches vielen Scannern angegliedert ist, ist die optische Zeichenerkennung, die weit verbreitete Schreibmaschinen- oder Druckschriften erkennt und in ASCII-Code umwandelt. Somit lassen sich ganze Seiten in Textdateien konvertieren und mit gebräuchlichen Textverarbeitungssystemen weiterverarbeiten. In letzter Zeit sind auch die ersten DTP-Scanner auf dem Markt erschienen, die es erlauben, farbige Abbildungen nicht in Grautönen, sondern in Farbe zu digitalisieren. Das ist eine der sich abzeichnenden Entwicklungsrichtungen. Die andere ist die Ausweitung der Fähigkeit, schriftgesetzten Text lesen zu können. Andere interessante Richtungen sind die Entwicklung von kleinen billigen Handscannern und die Verbindung von Scannern mit Fax-Software, um einen Scanner auch als Fax-Gerät zur Übertragung von Bildern über das Telefonnetz einsetzen zu können.

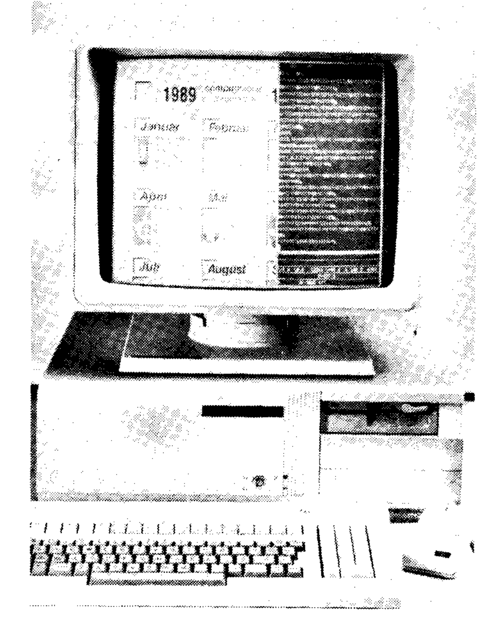
Bildschirme

Auf dem Bildschirm erfolgt beim DTP die ganze Arbeit der Zusammenfügung einer Seite des Dokuments. So wird man bald merken, daß die herkömmlichen PC-Bildschirme, unabhängig vom Grafikstandard, mit dem sie ausgerüstet sind, dafür nicht besonders gut geeignet sind. Auf einem 13-Zoll-Bildschirm läßt sich eine ganze Seite nicht im Detail darstellen, man benötigt ein häufiges Umschalten von reduzierter zu normaler oder meistens vergrößerter Darstellung. Das Editieren von Texten oder Grafiken kann zum Alptraum werden, da außerdem auch eine Verzerrung der Abbildungen (z. B. werden Kreise oft als Ellipsen dargestellt) stattfindet. Deshalb geht die Tendenz für DTP-Anwendungen zu großformatigen Bildschirmen mit hoher Auflösung. Damit werden Produktivität und Akkuratesse für DTP-Systeme zu einem großen Teil vom verwendeten Bildschirm bestimmt. Eine Untersuchung zeigte, daß etwa ein Drittel der Zeit bei der Herstellung einer druckfertigen Seite eingespart werden kann /2/. Nur mit einem großformatigen Bildschirm hoher Auflösung läßt sich der von den am meisten genutzten DTP-Softwarepaketen erhobene Anspruch „Was du (auf dem Bildschirm) siehst, ist was du (auf dem Drucker)



Den nach eigenen Angaben kleinsten Farbscanner der Welt präsentierte Sharp zur CeBIT '89. Der Handy-Farbscanner JX-100 soll besonders für nichtprofessionelle DTP-Anwender erschwinglich sein. Das Gerät scannt Vorlagen bis zur Größe von A6 bei einer Auflösung von 200 dpi.

erhältst" rechtfertigen (WYSIWYG – What you see is what you get). Bildschirme mit hoher Auflösung beginnen bei einer Bildschirmdiagonale von 16 Zoll und reichen gewöhnlich bis 19 Zoll, Hoch- oder Querformat. Die Auflösung variiert zwischen 1024 und 2880 Pixel horizontal und zwischen 600 und 1280 vertikal. Verglichen mit einer Auflösung von 640 × 320 für den EGA(Enhanced Grafics Adapter)-Standard von IBM ist das ein wesentlicher Fortschritt. Allerdings sind diese Bildschirme nur als Schwarzweiß-Bildschirme ausgelegt. Eine der Richtungen, die die Entwicklung von Bildschirmen mit hoher Auflösung nimmt, ist nicht so sehr die Erhöhung der Auflösung für die Textdarstellung, denn diese ist faktisch durch die verwendete Schriftgröße begrenzt (und wird kaum unter 6 typographischen Punkten liegen), sondern



Weniger für das gelegentliche „Publizieren am Schreibtisch" als vielmehr für professionelles Arbeiten im Satz- und Layoutbereich präsentierte die Compugraphic Deutschland GmbH kürzlich die Workstation Integrator 19/386 mit 80386-Prozessor, 19-Zoll-Bildschirm, 80-MByte-Festplatte und Unix-Betriebssystem. Damit ist es beispielsweise möglich, Echt-Schriften (WYSIWYG-Prinzip) auf der einen und die Befehle auf der anderen Seite des Bildschirms darzustellen, das heißt, interaktives, paralleles Arbeiten an der Analogdarstellung und im Befehlsprogramm. Die so gestalteten kompletten Seiten lassen sich wahlweise auf hochauflösende Laserbelichter oder Laserdrucker ausgeben.

eher die Erhöhung der Anzahl der abgebildeten unterschiedlichen Grautöne für Abbildungen, um diese entsprechend manipulieren zu können. Es zeichnet sich langsam ein Standard für diese Bildschirme ab, der bei 1280 × 960 oder 1280 × 1024 Pixel liegen wird. Mit der Einführung eines Standards würde sich die Zahl der gegenwärtigen Treiber und Steuereinschübe wesentlich verringern, und herkömmliche Textverarbeitungs-, Tabellen oder Grafikprogramme könnten mit einem Standardtreiber ausgerüstet werden. Die Preise für großformatige Bildschirme mit hoher Auflösung liegen bei dem drei- bis fünffachen Preis eines IBM VGA (Video Graphics Array)-Farbbildschirms oder dem sechs- bis zehnfachen Preis eines CGA(Color Graphic Adapter)-Farbbildschirms.

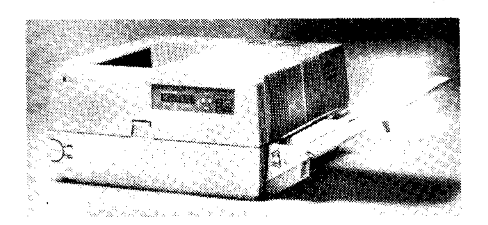
Maus

Bedingt durch die gleichzeitige Verwendung von Texten, Grafiken und Abbildungen in einem Dokument sind DTP-Softwarepakete mit einem grafischen Interface ausgestattet, welches die Nutzung einer Maus voraussetzt. Die Maus dient dabei zusammen mit dem Keyboard als Steuerungs- oder Eingabegerät für bestimmte Anwendungen, z. B. der Erzeugung mit der DTP-Software. Eine Maus gehört für neuere Apple-Macintosh-Computer bereits zur Standardausrüstung. Mit dem neuen OS/2-Betriebssystem für IBM-PCs wird die Maus auch für diese Computer zum Standard werden. Nutzer von Microsoft Windows oder Digital Research GEM, beides weit verbreitete grafische Interfaces, die auch in DTP-Software Verwendung finden, sind schon seit längerem an die Arbeit mit der Maus gewöhnt. Eine Maus besitzt eine Auflösung von 200 bis 320 Pixel pro Zoll und hat damit eine bessere Auflösung als herkömmliche Bildschirme. Es gibt Mäuse mit einer, zwei oder drei Drucktasten. Der eigentliche Unterschied zwischen verschiedenen Typen liegt in der Art der Übertragung der Bewegung. Hier gibt es die optische Maus, in der Lichteffekte, von optischen Sensoren registriert, in die Bewegung des Zeigers auf dem Bildschirm umgewandelt werden. Die mechanische Maus basiert darauf, daß ein Gummiball bewegt wird, der mechanische Sensoren veranlaßt, den Zeiger auf dem Bildschirm zu bewegen. Der Vorteil der optischen Maus liegt in ihrer Zuverlässigkeit und ihr Nachteil darin, daß sie eine Unterlage mit einem Raster und damit mehr Platz als eine mechanische Maus benötigt. Deren Nachteil besteht in der Verschmutzungsgefahr des Gummiballs, mit der ihre Leistungsfähigkeit abnimmt. Das führte zur Entwicklung der optomechanischen Maus, in der sich ein Gummiball bewegt, dessen Bewegungen dann mittels optischer Erkennungssysteme auf den Bildschirm übertragen werden. Die Maus kann in serieller (dabei wird der serielle Ausgang am PC genutzt) oder Bus-Ausführung (dafür wird ein Einschub benötigt) geliefert werden. Die Maus ist ein sehr billiges Eingabegerät und liegt meist unter dem Preis eines Keyboards; Treiber für die bekanntesten Softwarepakete werden in vielen Fällen von den Maus-Herstellern mitgeliefert.

Laserdrucker

Das am weitesten verbreitete Ausgabegerät in DTP-Arbeitsplätzen ist der Laserdrucker, der auch beste Ausgabequalität garantiert. Die Entwicklung dieser Drucker geht in die Richtung höherer Auflösung, geringer Preise,

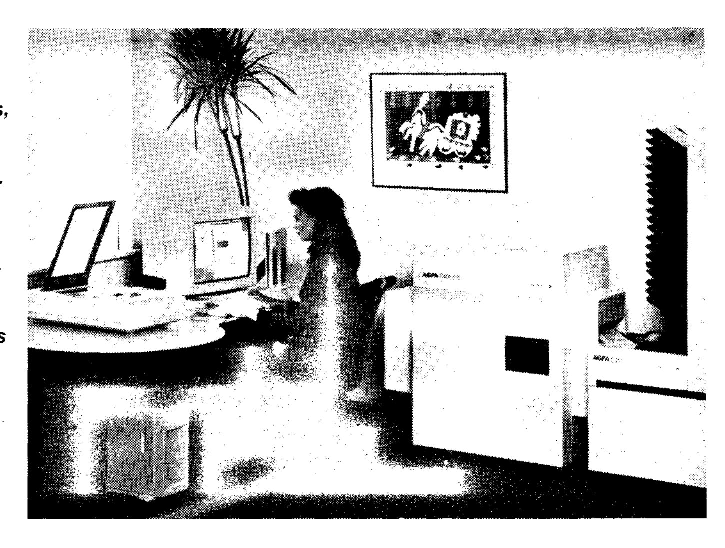
verbesserter Funktionstüchtigkeit und besserer Farbqualität, einschließlich Mehrfarbendruck. Damit sollen die Ansprüche insbesondere von DTP-Nutzern weiter befriedigt werden und die Fähigkeiten, die sich aus der Anwendung anderer Komponenten von DTP-Systemen ergeben, besser genutzt werden. Der Hauptgrund, warum Laserdrukker mehr und mehr Anwendung finden, ist die hohe Ausgabequalität. Danach kommen Funktionstüchtigkeit, Preis sowie Hard- und Softwarekompatibilität. Laserprinter starten heute bei einer Auflösung von 240 und reichen bis 600 Pixel je Zoll. Der Standard ist 300 Pixel je Zoll. Dies entspricht zwar Briefqualität, reicht aber noch nicht an Schriftsatzqualität heran, die bei etwa 600 Pixel je Zoll beginnt. Für diese Qualität bezahlt man einen hohen Preis, der vier- bis fünfmal höher als der von Standardlaserdruckern ist. Ein Faktor, der den Preis beeinflußt, ist der Preis für den Hauptspeicher, den der Laserdrucker benötigt. So besitzt ein Laserprinter mit 600 Pixel je Zoll Auflösung mindestens 4 MByte eigenen Hauptspeicher. Die meisten Laserprinter sind mit einer Reihe eigener Schriftsätze ausgestattet und simulieren verschiedene Druckstandards.



Ein typischer Vertreter der gegenwärtigen leistungsfähigen Tisch-Laserdrucker ist der neue PC Laser 6000/PS von RICOH. 300 dpi Auflösung, Druckgeschwindigkeit 6 Seiten/Minute, Motorola-68000-Mikroprozessor, 2 bis 4 MByte RAM sind seine Merkmale. Hervorgehoben wird die bei diesem Modell integrierte Seitenbeschreibungssprache PostScript, so daß die Arbeit mit bekannten DTP-Programmen möglich wird.

Für DTP hat sich eine eigene Seitenbeschreibungssprache herausgebildet, die mehr oder weniger zum Standard geworden ist – PostScript. Diese Sprache gestattet es, Text und Grafik gemeinsam auszugeben, wobei der Text ebenfalls im Grafikmodus gedruckt wird. Die gebräuchlichsten DTP-Softwarepakete besitzen einen PostScript-Treiber und senden die zu druckenden Dokumente im PostScript-Format zum Laserdrucker, der diese dann im eigenen Hauptspeicher aufbereitet. Dabei beeinflußt die Größe des Hauptspeichers wesentlich die Druckgeschwindigkeit mit. Laserprinter besitzen eine Reihe sich schnell abnutzender und anfälliger Teile. Sie bedürfen daher periodischer und aufmerksamer Wartung, des relativ häufigen Wechsels einzelner Teile und der ständigen Nachfüllung von Toner. Dadurch ergeben sich ständige hohe Nebenkosten, die bald die Anschaffungskosten übersteigen können. Der Preis für Laserdrucker mit 300 Pixel je Zoll Auflösung liegt bei dem Drei- bis Fünffachen des Preises für einen 24-Nadeldrucker.

Verschiedene Produzenten arbeiten an der Entwicklung von Farblaserprintern. Erste Modelle zeichnen sich vor allem durch einen hohen Preis aus und haben deshalb noch keine massenhafte Anwendung gefunden. Die Alternativen für farbige Druckausgabe sind Thermo- und Tintendruck, die den Hauptanteil am Farbdruckermarkt besitzen und noch eine Weile halten werden.



Als Electronic Publishing bezeichnet Agfa das Einsatzgebiet seines Systems Agfa Press, das bereits über den Bereich des Desktop Publishing hinausgeht. Basis ist demzufolge eine leistungsstarke Sun-Workstation 386i/150 mit Scanner, WYSIWYG-Bildschirm und LED-Drucker P 400 PS, der leistungsfähiger als ein herkömmlicher Laserdrucker ist. Beispielsweise hat er eine Auflösung von 400 dpi und eine Leistung von 18 Drucken pro Minute. Selbstverständlich ist auch die Ausgabe auf einen Laserbelichter möglich, der bis zu 2400 dpi auf Reprofilmen für den Offsetdruck liefert.

Software

Die bekanntesten DTP-Softwarepakete sind Aldus Pagemaker und Rank Xerox Ventura Publisher, deren Marktanteile relativ gleich sind. Beide gewährleisten den Import von Texten, erzeugt mit den bekannten Textverarbeitungssystemen, von Grafiken, die mit den gebräuchlichsten Grafikprogrammen erstellt wurden, oder Abbildungen, die digitalisiert wurden. Beide Pakete verwenden ein grafisches Interface; Pagemaker nutzt Microsoft Windows und Ventura basiert auf GEM. Der wesentlichste Unterschied besteht in der Behandlung der Texte. Während Pagemaker diese in das eigene Format integriert, beläßt Ventura den Text in seinem ursprünglichen Format, so daß dieser immer wieder mit dem verwendeten Textverarbeitungssystem verarbeitet werden kann.

Anwendungsmöglichkeiten

DTP besitzt ein weit gefächertes Anwendungsspektrum, welches von der Erzeugung von Speisekarten, Einladungen, Lebensläufen bis hin zu Projektdokumentationen, wissenschaftlichen Studien, Berichten, Bedienungsanleitungen, Broschüren usw. reicht. So produziert Volvo das Bedienungshandbuch für das Modell 740 mittels DTP und konnte dadurch wesentlich die Operativität durch die kurzfristige Berücksichtigung von Änderungen erhöhen. Dabei bedient man sich einer Software, die eine Brücke zu AutoCAD beinhaltet. Die Kosten für die Herstellung des Handbuches konnten erheblich gesenkt werden. Im Sekretariat der Vereinten Nationen wird das Layout der monatlichen „Betriebs"-Zeitung, genannt „Sekretariatsneuigkeiten", seit kurzem mittels DTP erstellt. Der Autor dieses Artikels erzeugte über zwanzig Grafiken für einen wissenschaftlichen Bericht, außerdem ein Handbuch für die Benutzung einer Datenbank mit 25 Seiten. Über dabei gewonnene Erfahrungen mit Ventura Publisher wird in einem weiteren Beitrag berichtet werden.

Literatur

/1/ PC Publishing, Vol. 3, No. 2, Februar 1988, S. 14

/2/ PC Publishing, Vol. 3 No. 5, Mai 1988, S. 20

TERMINE

Grundlehrgänge Forth-Programmierung

WER? KDT-Bezirksverband Suhl und Technische Hochschule Ilmenau
WANN? 4. bis 8. September 1989 (1. Lehrgang)
11. bis 15. September 1989 (2. Lehrgang)
WO? Rechentechnisches Kabinett der TH Ilmenau
WAS?
- Aufbau und Arbeitsweise von Forth
- Einführung in den Grundwortschatz von Forth-83
- Erarbeiten und Testen von Beispielprogrammen

WIE? Anmeldungen und Auskünfte beim KDT-Bezirksverband, Bereich Weiterbildung, PSF 510, Suhl, 6000; Tel. 22112

Dr. Finsterbusch

Kleines Lexikon der Mikrorechentechnik

K
wie Kelleroperation



Zeichnung: Dahmen

Logikanalyse des Mikroprozessors 8086/88

Frank Piepiorra, Gunnar Rothmann
Technische Universität Dresden,
Sektion Informationstechnik

Einführung

Für die Lösung zahlreicher Probleme, die bei der Entwicklung und beim Betrieb von Hard- und Softwarekomponenten von mikroprozessorgesteuerten Schaltungen und Geräten auftreten, haben sich Logikanalysatoren als vielseitig einsetzbare Meßgeräte bewährt. In /1/ und /2/ wurde darauf hingewiesen, daß für die Analyse eines integrierten Mikroprozessors der Einsatz eines angepaßten Adapters zwischen dem Meßobjekt und dem Logikanalysator notwendig ist. Diese als Hardware von Prozessorproben bezeichneten Anordnungen können passiv oder aktiv sein /2/, /3/. Um eine hohe Wirksamkeit des Logikanalysators garantieren zu können und im Interesse einer einfachen Betriebssoftware, ist es für viele Prozessoren günstiger, aktive Proben einzusetzen.
Anlehnend an /4/ wurde die Hardware einer Prozessorprobe für die Mikroprozessoren I8086 und I8088 so entwickelt, daß diese prinzipiell an jeden Logikanalysator, der folgende Kriterien erfüllt, anschließbar ist:
- Kanalbreite: 52 Kanäle (der Einsatz von Logikanalysatoren mit weniger Meßkanälen ist prinzipiell möglich, aber mit einem entsprechenden Informationsverlust verbunden)
- Abtastfrequenz: 10 MHz bei externem Takt über einen externen Takteingang.

Daraus ist ersichtlich, daß der Logikanalysator in Zusammenarbeit mit der Prozessorprobe in der Betriebsart Zustandsanalyse /5/ betrieben wird. Mikrorechnergesteuerte Logikanalysatoren, welche eine solche Betriebsart zulassen, verfügen auch über eine entsprechende Software zur Anzeige der aufgezeichneten Meßdaten in hexadezimaler oder binärer Form. Damit wird schon eine einfache Auswertung der (mit Hilfe der Hardware einer Prozessorprobe) gewonnenen Meßdaten, zur Untersuchung von Hard- und Software durch die Anzeige einer vollständigen oder unvollständigen Programmspur oder der Busaktivitäten des Mikroprozessors, in Tabellenform möglich /2/. Weitere leistungsfähigere Auswerteverfahren, beispielsweise eine Disassemblierung der Programmspur oder eine Softwareleistungsanalyse, erfordern die schon erwähnte angepaßte Probenbetriebssoftware.

Betriebsarten der Prozessorprobe

Als eigenständige Vorsatzkomponente für einen Logikanalysator besteht die Hauptaufgabe der Hardware einer Prozessorprobe darin, die an den Prozessorpins des Meßobjektes anliegenden Informationen geeignet zu interpretieren und zu qualifizieren, daraus einen definierten Abtasttakt für den nachfolgenden Logikanalysator zu generieren und die qualifizierten Informationen über die Funktion des Meßobjektes zum Abtastzeitpunkt dem Logikanalysator (in geeigneter Form) zur Verfügung zu stellen. Betrachtet man die Arbeitsweise eines beliebigen nach der Von-Neumann-Architektur aufgebauten Mikroprozessors, so können aus den an den Prozessorpins anliegenden
- Status- und/oder Steuersignalinformationen
- physischen Adreßinformationen
- Daten- und Operandeninformationen

umfangreiche Schlußfolgerungen über die Arbeit des Prozessors in einem System gezogen werden. Die aktive Prozessorprobe erkennt und verarbeitet vorrangig diese Informationen. Im vorliegenden Fall gilt das für die Prozessoren I8086 und I8088, die sowohl im Minimum- als auch im Maximummodus betrieben werden können.
Der Prozessor I8086 vereinigt zwei Spezialprozessoren auf einem Chip (siehe Bild 1). Zum einen die *Execution-Unit* (EU) und zum anderen die *Bus Interface Unit* (BIU).

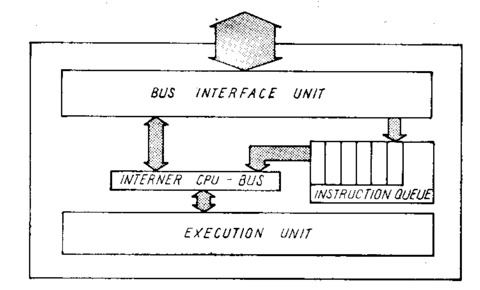


***Bild 1** Vereinfachtes Blockschaltbild des Prozessors I8086*

Die *Execution Unit* verfügt über alle Komponenten eines herkömmlichen Mikroprozessors zur Befehlsausführung und Operandenmanipulation. Der wesentliche Unterschied zu vorhergehenden Typen (z. B. I8008, I8080) besteht darin, daß die EU keinen Zugriff auf den externen Bus besitzt. Diese Art der Kommunikation wird von der *Bus Interface Unit* verwaltet. Sie organisiert alle erforderlichen Buszugriffe. Eine interne Warteschlange (*Instruction Queue*) wird durch die BIU entsprechend dem Pipeline-Prinzip mit Befehlen, die auf den sequentiell folgenden Adressen stehen, gefüllt. Aus dieser Struktur ergibt sich, daß bei einer Programmverzweigung nicht alle in der Warteschlange abgespeicherten Befehle auch wirklich abgearbeitet werden /6/. Aus dieser Aussage lassen sich für die Prozessorprobe zwei Betriebsarten ableiten:

Busmodus

Die Abarbeitung von Befehlen im Mikroprozessor wird durch eine definierte Aneinanderreihung von Buszyklen organisiert. Welche verschiedenen Arten von Buszyklen es beim Prozessor I8086 gibt, zeigt Tafel 1.
Die durch die BIU generierten Buszyklen ermöglichen die Kommunikation des Mikroprozessors mit seiner Hardwareumgebung. Wünscht ein Nutzer die Aufzeichnung der Informationen, die während der Zeit eines Buszyklus verarbeitet werden, so wird am Ende des jeweiligen Zyklus in der Prozessorprobenhardware ein Abtasttakt generiert. Mit diesem Abtasttakt werden alle relevanten, meist in Zeitteilung vorliegenden Informationen, nachdem sie in der Hardware der Prozessorprobe in die Raumteilung überführt wurden, parallel in den als Zustandsanalysator arbeitenden Logikanalysator übernommen.

***Tafel 1** Busaktivitäten des 8086/8088 (nach /7/)*

Belegung der Statussignale			Buszyklus
/S0	/S1	/S2	
0 (Low)	0	0	INTERRUPT ACKNOWLEDGE
0	0	1	READ I/O PORT
0	1	0	WRITE I/O PORT
0	1	1	HALT
1 (High)	0	0	CODE ACCESS
1	0	1	READ MEMORY
1	1	0	WRITE MEMORY
1	1	1	PASSIV (NO BUS CYCLE)

Die Prozessorprobe ermöglicht es, die Auswahlkriterien für die interessierenden Buszyklen in Abhängigkeit vom Verlauf des Meßvorganges dynamisch zu verändern. Als Ergebnis dieser Analyse erhält man ein Protokoll aller Aktivitäten der BIU, wie sie sich aus dem Verhalten der EU ableiten. Es wurden auch die Buszyklen erfaßt, die die BIU zum Füllen ihrer internen Warteschlange einleitet, unabhängig davon, ob der eingeschriebene Operationscode dann auch wirklich abgearbeitet wird. Durch die Arbeitsweise des Prozessors kann es dabei zu einer Verschachtelung der Buszugriffe, die durch die EU oder die BIU initiiert wurden, kommen. Diese Betriebsart eignet sich sowohl zur Untersuchung der Hardware des Meßobjektes, als auch zur Funktionsüberprüfung der Ressourcenverwaltung durch die Software eines Systems.

Befehlsmodus

Diese Betriebsart erlaubt die Aufzeichnung einer Programmspur als eine sequentielle Folge aller abgearbeiteten Befehle und einer eventuell darauf folgenden Aktivität des Prozessors auf dem Bus (z. B. Operandentransfers). Alle durch die EU nicht abgearbeiteten Befehle, die aber durch die BIU in die Warteschlange eingetragen worden sind, werden erkannt und in der Prozessorprobenhardware ausgesondert. Dadurch wird erreicht, daß
- die Triggerung des Logikanalysators ausschließlich auf ausgeführte Programmabschnitte reagiert
- die Auswertung der Meßdaten übersichtlich und einfach wird
- die Überführung in andere Darstellungsformen der Meßwerte (z. B. mnemonische Darstellung) durch spezielle Betriebssoftware wesentlich erleichtert werden kann
- der Meßwertspeicher des Logikanalysators weitgehend von irrelevanten Eintragungen befreit wird.

Diese Art der Meßwerterfassung unterstützt besonders die Methoden zur Softwareuntersuchung im Meßobjekt.
Werden beide Betriebsarten der Prozessorprobe miteinander kombiniert, so lassen sich

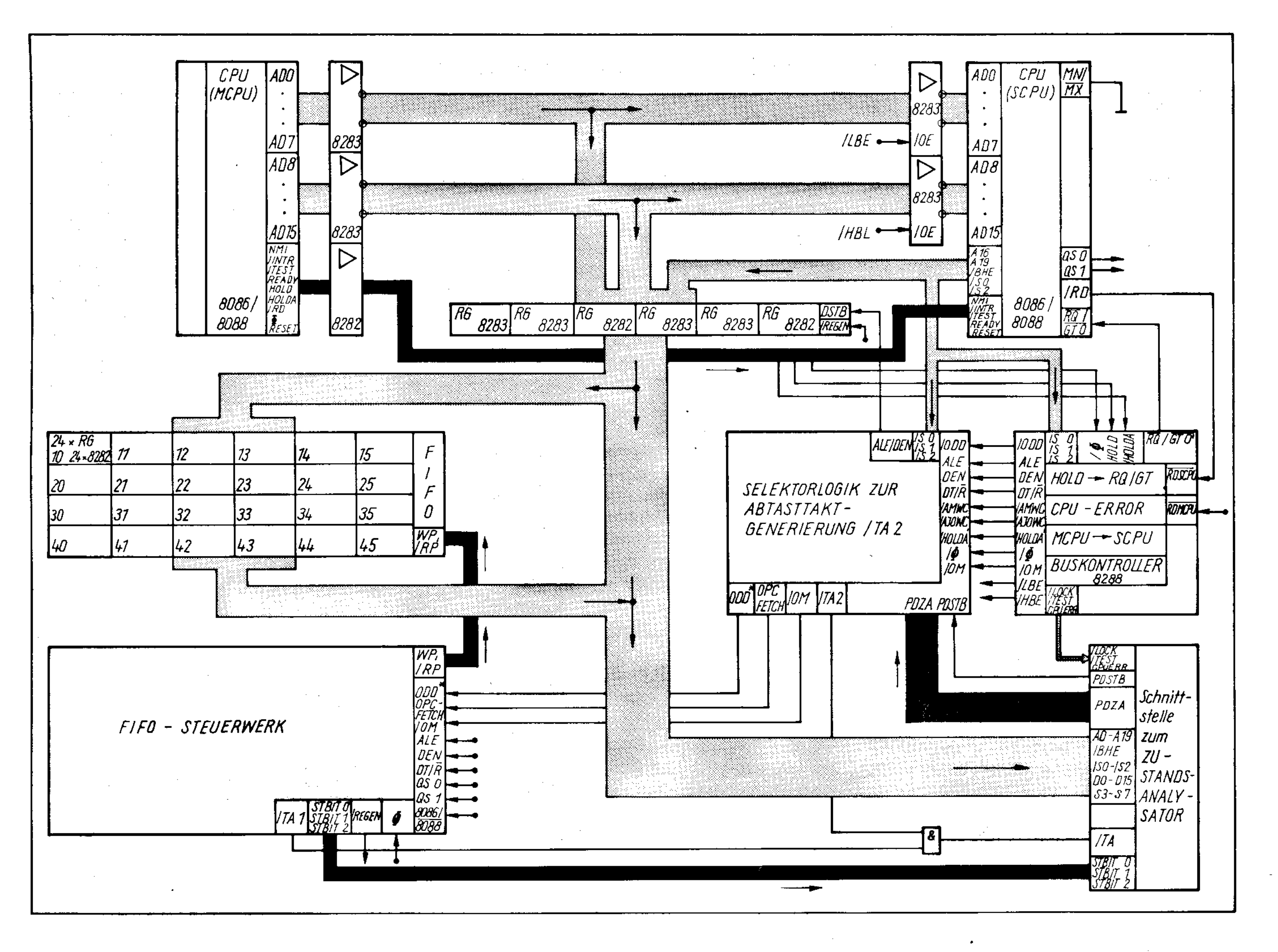


Bild 2 Blockschaltbild der Prozessorprobe für I8086/8088

sehr leicht die kausalen Bezüge zwischen den registrierten Veränderungen von Systemressourcen und den diese auslösenden Softwarekomponenten herstellen.

Konzeption der Hardware

Das Blockschaltbild der Hardware der Prozessorprobe zeigt Bild 2. An Hand der dargestellten Komponenten wird im folgenden eine grobe Erläuterung der verschiedenen Funktionen gegeben und auf Modifikationsmöglichkeiten hingewiesen. Diese Konzeption erlaubt es, abhängig von den Meßbedingungen und -zielen, auf einzelne Funktionen zu verzichten und den doch erheblichen schaltungstechnischen Gesamtaufwand anwenderspezifisch zu senken.

Allein aus den an den Prozessorpins des Meßobjektes vorliegenden Informationen ist es nicht in jedem Falle möglich, beide vorgestellten Betriebsarten zu realisieren. Dazu ist es erforderlich, die interne Kommunikation zwischen der EU und der BIU für die Verwaltung der internen Warteschlange auszuwerten. Eine im Maximummodus betriebene CPU liefert über zwei Statuspins (QS0, QS1) alle dazu notwendigen Informationen. Um auch bei einer im Minimummodus betriebenen CPU diese Angaben erhalten zu können, wird eine parallel arbeitende CPU im Maximummodus mit völlig gleichen Daten versorgt. Die dabei auftretenden und in /4/ erwähnten Probleme wurden gelöst, und die synchrone Arbeitsweise der beiden Prozessoren wird während des gesamten Meßprozesses kontrolliert. Arbeitet das Meßobjekt ausschließlich im Maximummodus, kann auf die Parallel-CPU verzichtet werden.

Tafel 2 gibt einen Überblick über die Struktur der von der Prozessorprobe gelieferten Meßdaten.

Realisierung des Busmodus

Die Schaltung besteht aus der Selektorlogik zur Abtasttaktgenerierung für den Logikanalysator und einer Busregisterlogik. Zusätzlich werden hier Signale erzeugt, die einen Befehlsanfang und einen Zugriff auf eine ungerade Adresse anzeigen und die Auswertung der Meßdaten im Logikanalysator erleichtern helfen sollen.

Alle in Tafel 1 aufgelisteten Buszyklen lassen sich disjunktiv verknüpft zur Gewinnung der Meßdaten heranzuziehen. Mit geeigneten Takten werden alle benötigten Informationen während der ausgewählten Buszyklen in die Busregister übernommen und raumgeteilt. Am Ende eines Buszyklus erfolgt dann die Übergabe an den Logikanalysator durch das Generieren eines externen Taktes.

Realisierung des Befehlsmodus

Für diese Betriebsart wurde extern die Schnittstelle der Datenübergabe der Befehlscodes aus der Warteschlange der BIU an die EU nachgebildet. Dadurch läßt sich zeitgleich mit dem Zugriff der EU auf die Warteschlange der abzuarbeitende Befehlscode an den Logikanalysator übertragen. Voraussetzung dafür ist ein Speicher, mit dem sich transient

Tafel 2 Struktur der Meßdaten

Meßkanäle des Logikanalysators	Meßwert	Erläuterung
1–20 21	A0–A19 /BHE	physische Adresse mit Informationen über die angesprochenen Speicherbänke
22–24	/S0–/S2	Statuswort der CPU im Maximummodus zur Charakterisierung der Art des Buszugriffs
25–40	D0–D15	Operanden, Befehle, Daten, die über den 16-Bit-Daten-/Adreßbus übertragen wurden
41–45	S3–S7	durch die CPU ausgegebenes Statuswort, repräsentiert einen Teil des PSW der CPU
46–48	STBIT0–STBIT2	durch die Prozessorprobe generierte Statussignale zur Interpretation der Befehlsbytes
49	/LOOK	Zustand des LOOK-Pins zum Zeitpunkt der Abtastung
50	EU-Zugriff	unterscheidet Befehlscodes von Operanden
51	CPU-Error	Überwachung der synchronen Arbeitsweise der Probe
52	/TEST	Zustand des TEST-Pins zum Zeitpunkt der Abtastung

die BIU-Aktivitäten protokollieren lassen. Dieser Speicher, mit logischer FIFO-Struktur, besitzt für die Prozessorprobe vier Ebenen mit je 48 Bit Speicherkapazität. Die Verwaltung übernimmt ein Steuerwerk, das folgende Funktionen organisiert:

1. Beschreiben des Speichers durch BIU-Aktivitäten (FIFO-WR-Logik)

Die FIFO-WR-Logik generiert nach Ablauf eines Buszyklus, der einen Eintrag in die interne Warteschlange des Prozessors zur Folge hat, einen Schreibimpuls (WR), der die Übernahme der Informationen über die Prozessoraktivität in die aktuelle FIFO-Ebene zur Folge hat.

2. Lesen des Speichers durch den Logikzustandsanalysator (FIFO-RD-Logik)

Unter der FIFO-RD-Logik sind vereinigt:
– eine Logik zum Adressieren und Auslesen einer FIFO-Ebene; sie organisiert, daß die aktuellen Meßdaten zum Abtastzeitpunkt an den Meßkanälen des Logikanalysators stabil anliegen.
– eine Logik zum Generieren der Statussignale STBIT0 – STBIT2; diese Statussignale dienen ebenfalls dem Kennzeichnen der Meßdaten zur Unterstützung ihrer Interpretation bei der Auswertung. Es gilt folgende Zuordnung:

STBIT0
– aktiv: Beide Bytes (D0–D7 und D8–D15) des an den Logikanalysator übertragenen Meßwertes sind gültig, das heißt, beide Bytes wurden durch die EU aus der Warteschlange geholt und abgearbeitet.
– inaktiv: Nur das NWT-Byte (D0–D7) ist gültig. Die Warteschlange wurde wegen einer Programmverzweigung gelöscht. Gleichzeitig ist das STBIT2 ungültig.

STBIT1
– aktiv: Das NWT-Byte ist ein FIRST BYTE OF OPC.
– inaktiv: Das NWT-Byte ist ein SUBSEQUENT BYTE OF OPC.

STBIT2
Kennzeichnet analog zur Funktion von STBIT1 die Spezifikation des HWT-Bytes (D8–D15).

3. Löschen des FIFO

Das Löschen der Warteschlange in der CPU bei einer beliebigen Programmverzweigung wird im externen FIFO nachvollzogen.

4. Generieren eines Abtasttaktes für den Logikanalysator

Mit dieser Logik wird ein entsprechender Abtasttakt als Reaktion auf eine EU-Aktivität nach Bereitstellen der Daten an den Meßkanälen für den Logikanalysator erzeugt.
Für das Aufzeichnen einer Operation der CPU als Ergebnis der Abarbeitung eines Befehls (z. B. Datentransport zum Speicher oder zum Ausgabetor) wird die Logik für den Busmodus hinzugezogen. Da auch im Fall des „fast" gleichzeitigen Zugriffs der EU auf die interne Warteschlange und der BIU auf Systemressourcen bei der Abarbeitung eines Programms alle dabei verarbeiteten Informationen an den Logikanalysator übertragen werden müssen, ergibt sich für den Logikanalysator eine maximale Abtastfrequenz, die über der Taktfrequenz des Mikroprozessors liegt.
Während für alle Schaltungskomponenten konventionelle Lösungen mit Hilfe einfacher kombinatorischer und sequentieller Automaten angewendet wurden, macht es sich erforderlich, die komplexe FIFO-RD-Logik mit einem schnellen sequentiellen PROM-Steuerwerk aufzubauen. Dadurch lassen sich, neben der Minimierung des schaltungstechnischen Aufwandes und einer hohen Reaktionsgeschwindigkeit, einheitliche Signalverzögerungszeiten erzielen.

Ausblick

Die zur Analyse am Mikroprozessor I8086 gemachten Aussagen lassen sich sinngemäß auf den I8088 übertragen. Auf spezielle Einzelheiten soll nicht eingegangen werden.

Vor Beginn der Messung nimmt der Nutzer eine Anpassung der Prozessorprobe an sein Meßobjekt vor. Das betrifft die Auswahl des Prozessortyps und die Betriebsartenauswahl der CPU sowie die Initialisierung der Prozessorprobe. Diese Angaben werden während der ganzen Messung konstant gehalten.
Die Auswahl der aufzuzeichnenden Buszyklen kann während der Messung dynamisch verändert werden. Manche Logikanalysatoren bieten dazu die Möglichkeit der Ausgabe eines Steuerwortes in Abhängigkeit vom Erkennen einer bestimmten Datenfolge auf den Meßkanälen. Welche von den durch die Prozessorprobe angebotenen Daten letzlich im Meßwertspeicher des Logikanalysators wiederzufinden sind, hängt zunehmend von den Fähigkeiten der zum Einsatz gelangenden Logikanalysatoren zur weiteren Datenreduktion und -selektion ab.

Literatur

/1/ Meusel, K.-H.; Piepiorra, F.: Softwareanalyse mit Logikanalysesystem LAS 20. Radio, Ferns., Elektron., Berlin 35 (1986) 2, S. 71
/2/ Piepiorra, F.: Logikanalyse an mikrorechnergesteuerten Geräten. Nachrichtentech., Elektron., Berlin 37 (1987) 1, S. 17
/3/ van Eijkelenburg, H.; Machnik, K.: Disassembler vereinfacht Logikanalyse. Elektronik, München (1984) 15, S. 66
/4/ Grebenstein, F.: Logikanalyse an 8086-Systemen, Elektronik, München (1981) 23, S. 87
/5/ Götze, B.; Meusel, K.-H.; Nicklisch, G.: Logikanalysator LA 32/20. Radio, Ferns., Elektron., Berlin 34 (1985) 10 u. 11, S. 626 u. 719
/6/ INTEL: iAPX 86/88. iAPX 186/188. User's Manual – Programmers Reference. Intel 1986
/7/ INTEL: Microsystem Components Handbook, Microprocessors And Peripherals. Volume 1, Intel 1985

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Informationstechnik, B2, Mommsenstraße 13, Dresden, 8027; Tel. 4632118

Wo stehen wir bei der Softwareproduktion?

Mit der breitenwirksamen Nutzung moderner Personalcomputer, hochleistungsfähiger 32-Bit-Rechentechnik sowie arbeitsplatznaher CAD/CAM-Systeme und den geschaffenen technischen Voraussetzungen, um Datennetze zu nutzen, wurde das Anwendungsniveau der modernen Rechentechnik und CAD/CAM-Technologie bedeutend erhöht. Damit hat sich Software als wirksamste Form, in der wissenschaftlich-technisches Wissen und die gesamte Erfahrung über komplexe Informations- und Kommunikationsprozesse mit Hilfe der modernen Rechentechnik direkt nutzbar gemacht werden, zu einem der Faktoren herausgebildet, die das technologische Niveau der DDR als entwickeltes Industrieland bestimmen.
Im Jahr 1987 wurde – bei einem durchschnittlichen Zuwachs von 25 Prozent zwischen den Jahren 1986 und 1988 – der Betrag von einer Milliarde Mark für das Wertvolumen der Softwareherstellung überschritten. Im gleichen Zeitraum nahm das Wertvolumen der für den eigenen Bedarf, d. h. für die Rationalisierung der betrieblichen Prozesse in der Leitung und Verwaltung sowie der technisch-technologischen Prozesse der Produktion, hergestellten Softwarelösungen um etwa 38 Prozent pro Jahr zu. Die Effektivität der Softwareherstellung wurde seit 1986 jährlich um 6 Prozent durch die Nutzung moderner Softwarewerkzeuge und -herstellungstechnologien gesteigert.
Die für die nächsten Jahre zu erwartende Entwicklung des Softwarebedarfs und -herstellungspotentials macht deutlich, daß allen produktivitätserhöhenden Maßnahmen und einer weiteren Forcierung der Mehrfachnutzung eine entscheidende Rolle für die Dekkung des Bedarfs an Software zukommt.
Für 70000 CAD/CAM-Arbeitsstationen, die bis 1988 zum Einsatz gelangten, wurde leistungsfähige Basissoftware und CAD/CAM-Anwendungssoftware hergestellt und in die produktive Nutzung überführt. Allein damit ist für weit über 20000 Werktätige in den Bereichen der Leitung und Planung, der Forschung und Entwicklung und in der Produktion, also unmittelbar dort, wo das Tempo des breitenwirksamen Einsatzes der CAD/CAM-Schlüsseltechnologie bestimmt wird, ein qualitativ neues Niveau für eine höhere Effektivität der eigenen Arbeit und die Rationalisierung komplexer arbeitsteiliger Prozesse geschaffen worden. Analog zu den internationalen Erfahrungen beträgt der Anteil am Gesamtaufwand, der für anspruchsvolle und komplexe Lösungen durch die Anwender selbst zu realisieren ist, etwa 50 bis 70 Prozent.
Um die Anwender bei der effektiven Entwicklung eigener Lösungen zu unterstützen und die Nach- und Mehrfachnutzung zu fördern sowie verbindliche Regelungen zur Arbeitsteilung bei der Herstellung von Software zu schaffen, wurden 1986 die Anordnung über die Informations- und Beratungsleistungen zur Entwicklung, Produktion und Mehrfachnutzung von Software in der DDR (GBl. Teil I Nr. 9/1986) und die Anordnung über die Planung, Bilanzierung und Abrechnung von Software (GBl. Teil I, Nr. 4 1986) in Kraft

gesetzt. In Auswertung der bisher bei der Herstellung von Software gesammelten Erfahrungen hat der Ministerrat der DDR im vergangenen Jahr Maßnahmen zur weiteren Vervollkommnung der zentralen Leitung, Planung und Bilanzierung der Softwareherstellung auf der Basis eines abgestimmten strategischen Konzeptes der Entwicklung von Hard- und Software beschlossen.
Eine konzentrierte und konsequente Leitungstätigkeit spiegelt sich im hohen anwendungstechnischen Niveau der von den Bereichen Werkzeug- und Verarbeitungsmaschinenbau, Elektrotechnik und Elektronik, Chemische Industrie, Land-, Forst- und Nahrungsgüterwirtschaft, Bezirksgeleitete Industrie und Lebensmittelindustrie sowie Bauwesen und Erzbergbau, Metallurgie und Kali zum Einsatz gebrachten Anwendungslösungen wider. Hier hat sich die Bildung von Leiteinrichtungen, zentralisierten Koordinierungsorganen und konzentrierten leistungsfähigen Softwarekapazitäten, verbunden mit einer straffen Leitung der Prozesse, als Voraussetzung für eine produktive arbeitsteilige Herstellung und effektive Nutzung von Software bewährt.
Neben der weiteren Entwicklung dezentraler Kapazitäten der Softwareherstellung gewinnt der verstärkte Ausbau leistungsfähiger zentralisierter Softwarekapazitäten für die wirksame Anwendung ingenieurmäßiger Methoden der Softwareentwicklung einschließlich moderner Softwarewerkzeuge und effektiver Softwareentwicklungstechnologien wesentliche Bedeutung. So verfügen zum Beispiel Kombinate und Einrichtungen der Bereiche Elektrotechnik und Elektronik, Chemische Industrie, Handel und Versorgung sowie Verkehrswesen, in denen Software auf der Basis der Prinzipien der wirtschaftlichen Rechnungsführung hergestellt wird, über ein fortgeschrittenes Niveau der Herstellung von Anwendungssoftware. Dies ist durch einen hohen Anteil mehrfachnutzbarer Software und ein hohes Entwicklungstempo gekennzeichnet. Das zeigt, daß diejenigen Generaldirektoren gut beraten sind, die den weiteren Ausbau zentraler und leistungsfähiger Softwarekapazitäten in den Kombinaten unter ihrer Leitung konsequent fortsetzen und damit einen Weg beschreiten, der bei der weiteren Konzentration auf die Hauptlinien der Herstellung von Anwendungssoftware eine stabile Deckung des Softwarebedarfs gewährleisten wird.

Karl Nendel, Staatssekretär im Ministerium für Elektrotechnik und Elektronik

Umleitung für SUBMIT ist schneller

Frank Isekeit
Deutsche Reichsbahn, Ingenieurbüro für Rationalisierung des Eisenbahnbaues

Das im RAM angelegte logische Laufwerk E: des Arbeitsplatzcomputers A 7100/A 7150 bringt insbesondere bei der Arbeit mit solchen Programmen erhebliche Geschwindigkeitsvorteile, die eine große Anzahl von Diskettenzugriffen erfordern. Deshalb ist es sinnvoll, beim Systemstart die erforderlichen Dateien auf das Laufwerk E: zu kopieren und dieses dann als aktuelles Laufwerk zu wählen.
Eine besonders anwenderfreundliche und deshalb für Bediener ohne EDV-Kenntnisse zu empfehlende Möglichkeit dazu würde das Stapelverarbeitungsprogramm SUBMIT bieten, dessen Aufruf vorteilhaft in Form eines Kaltstartkommandos erfolgen kann /1/, /2/. Dabei zeigt sich aber ein Nachteil: Erscheint in der Kommandofolge ein Befehl zum Umschalten des aktuellen Laufwerkes auf E:, so wird dieser zwar abgearbeitet, alle danach folgenden Befehle aber ignoriert.
Für dieses Verhalten gibt es eine einfache Erklärung. Das Dienstprogramm SUBMIT wandelt die in der SUBMIT-Datei gespeicherte Kommandofolge in ein modifiziertes Format um und legt sie in der Arbeitsdatei $$$.SUB auf dem aktuellen Laufwerk ab /3/. Das ist nach dem Systemstart immer eines der Diskettenlaufwerke. Das Betriebssystem sucht nun auf dem aktuellen Laufwerk die Arbeitsdatei und arbeitet sie befehlsweise ab. Ein Wechsel des aktuellen Laufwerkes führt folglich zwangsläufig zum Abbruch der Abarbeitung.
Zur Lösung dieses Problems wäre es denkbar, ein Hilfsprogramm zu schaffen, das die Befehle zum Laufwerkswechsel auf E: und zum Aufruf des gewünschten Programmes direkt in den Tastatureingabepuffer im BIOS einschreibt (Realisierung der Funktionen der Tastatur-Interruptroutine). Dieses Programm könnte man als letzten Befehl in der SUBMIT-Datei aufrufen.
Ein solches Programm wurde praktisch erprobt und damit die prinzipielle Möglichkeit dieser Vorgehensweise bewiesen. Der entscheidende Nachteil der Methode besteht aber darin, daß sich die Adressen des Tastatureingabepuffers und der zu dessen Verwaltung erforderlichen Zeiger bei den Versionen 2.2 und 3.0 des Betriebssystems SCP 1700 um ein Byte unterscheiden und das Hilfsprogramm damit nicht universell einsetzbar ist.
Eine weitere Lösungsmöglichkeit besteht darin, das Dienstprogramm SUBMIT so zu verändern, daß es die Arbeitsdatei auf dem Laufwerk E: erstellt und vor Rückgabe der Steuerung an das Betriebssystem auch vom Kaltstartlaufwerk auf E: umschaltet. Damit entfällt die Notwendigkeit des Laufwerkswechsels während der Abarbeitung der Arbeitsdatei.
Für die praktische Realisierung wurde dieser Variante der Vorzug gegeben. Neben der Lösung des eingangs genannten Problems ergeben sich dabei zwei weitere Vorteile:

- Es können problemlos mehrere Stapelbefehle mit dem aktuellen Laufwerk E: abgearbeitet werden. Bei Nutzung eines Textverarbeitungssystems können so beispielsweise die bearbeiteten Dateien ohne Zutun des Bedieners bei Beendigung der Arbeit wieder auf eines der Diskettenlaufwerke zurückkopiert werden.
- Da der Rechner jeden abzuarbeitenden Stapelbefehl einzeln aus der Arbeitsdatei liest, treten durch die notwendigen Positioniervorgänge bei Diskettenlaufwerken relativ lange Verzögerungen auf, die bei der Arbeit vom Laufwerk E: (weitgehend) entfallen. Bei einem dazu durchgeführten Versuch konnte durch Abarbeitung vom Laufwerk E: im Vergleich zur Arbeit mit Laufwerk A: die Bearbeitungszeit von 55 s auf 45 s gesenkt werden.

Die Änderung kann mit Hilfe der Dienstprogramme DDT86 oder POWER erfolgen. An dieser Stelle sei jedoch nur die Vorgehensweise unter POWER dargestellt.
Zunächst wird die Datei SUBMIT.CMD in den Hauptspeicher geladen:

LOAD SUBMIT.CMD 4000<CR>

Bild 1 zeigt die mit Hilfe der *DS*-Funktion zu modifizierenden Speicheradressen jeweils mit dem alten und neuen Inhalt.

```
Adr.            Inhalt    zu ändern in

2081:409B         19          ØE
     409C         E8          B2
     409D         3E          Ø4
     409E         ØØ          E8
     409F         A2          3C

2081:40A1         Ø1          9Ø

2081:40AE         ØA          ØC
     40AF         Ø6          Ø4
     40BØ         ØØ          9Ø
     40B1         Ø1          9Ø

2081:45C9         ØØ          Ø5
```

***Bild 1** Unter POWER auszuführende Änderungen*

Nach Ausführung der Änderung wird die Datei auf die Diskette zurückgeschrieben:
SAVE name.CMD 4000 31 <CR>
Für name kann wieder SUBMIT angegeben werden. Im Interesse der eindeutigen Unterscheidung von der Originaldatei sollte man aber einen anderen Dateinamen wählen.
Das Dienstprogramm SUBMIT mit der genannten Änderung wurde im Rahmen eines größeren REDABAS-Programmpaketes eingesetzt und hat sich dabei sowohl am A 7100 als auch am A 7150 unter SCP 1700 bewährt.

Anmerkung
Werden mehrere Dateien, die sich nicht durch Dateigruppenspezifikationen zusammenfassen lassen, mit PIP nach E: kopiert, so ist es zweckmäßig, zunächst die Datei PIP.CMD zu kopieren und anschließend von E: aufzurufen, was ebenfalls die Abarbeitungszeit verkürzt.

Literatur
/1/ Lennartz, M.: Veränderungen des SCP 1700. Mikroprozessortechnik 2 (1988) 4, S. 124
/2/ Herse, M.; Isekeit, F.: Änderungen am Betriebssystem SCP 1700. Mikroprozessortechnik 2 (1988) 6, S. 181
/3/ Arbeitsplatzcomputer A 7100, Anleitung für den Bediener. Teil „Dienstprogramme", S. 66

✉ KONTAKT ☏
Deutsche Reichsbahn, Ingenieurbüro für Rationalisierung des Eisenbahnbaus, Bereich Brandenburg, Uferstraße, Brandenburg-Kirchmöser, 1802; Tel. 27 64 69

Externe Unterprogramme in Turbo-Pascal

Christian Hanisch, Berlin

Mit diesem Beitrag beenden wir unsere in MP 10/88 begonnene Artikelfolge zur hardwarenahen Programmierung, die in den Heften 11/88 und 3/89 fortgesetzt wurde.

Turbo-Pascal für CP/M-kompatible Betriebssysteme ist heute neben Basic-80 die dominierende Programmiersprache für Applikationen breiter Nutzerschichten in allen Branchen geworden. Bei Turbo-Pascal fällt im Unterschied zu Programmiersprachen wie Fortran, C oder Assembler das Fehlen eines Linkers auf. Im folgenden soll gezeigt werden, wie dieser *Mangel* sich methodisch positiv auf das Problem der dynamischen Einbindung externer Assembler-Unterprogramme in Turbo-Pascal-Programme auswirkt.
Im Turbo-Pascal-Programmiersystem kann über die O)ption E)nd address: C100 die höchste vom Pascal-Programm nutzbare TPA-Adresse niedriger als die aktuelle BDOS-Beginn-Adresse festgesetzt werden. Der dadurch freigehaltene Bereich kann unter anderem für sogenannte EXTERNAL-Unterprogramme genutzt werden. Diese externen Unterprogramme in Turbo-Pascal sind auf feste TPA-Adresse gelinkte und geladene Maschinencodeprogramme beliebiger konventioneller Programmiersprachen (meistens in Assemblersprache). In diesem Falle ist der Assembler-INLINE-Code als Bestandteil eines Turbo-Pascal-Quelltextes (siehe PMLINK /1/ oder INLASS /2/) eine alternative Methode zur externen Unterprogrammtechnik.
Da EXTERNAL-Unterprogramme als COM-Dateien auf eine feste von 100H verschiedene TPA-Adresse gelinkt und geladen werden, ist die Programmverbindungs-Schnittstelle unabhängig von der zugrundeliegenden Programmiersprache nur durch Parameterübergabekonventionen bedingt, denen sich die gerufenen externen Unterprogramme gemäß der Ruf-Konvention für externe Unterprogramme vom Turbo-Pascal-System aus fügen müssen. Die Programmverbindung von Turbo-Pascal mit anderen Programmiersprachen erfolgt im Gegensatz zu herkömmlichen Programmiersprachen nicht auf der Ebene von REL-Dateien und -Bibliotheken vermittels eines Linkers, sondern auf der Ebene von COM-Dateien über feste TPA-Adressen und ausführbaren Maschinencode. Diese Tatsache drängt zu einer Methodik dynamisch nachladbarer externer Unterprogramme beliebiger konventioneller Programmiersprachen auch in einer dynamischen Überlagerungsstruktur (external OVERLAY).
Mit dem sogenannten Run-Time-Lader, einer Turbo-Pascal-Prozedur XPOLADER.INC (siehe Bild 1), können beliebige externe Unterprogramme, die vorab als OVX-Dateien auf Diskette abgespeichert wurden, dynamisch auf die geplante TPA-Adresse geladen werden. Als Beispiel ist im Bild 2 mit dem Turbo-Pascal-Programm SPACETST.PAS der Einsatz des Run-Time-Laders in Form der PROCEDURE XPOLADER zum dynamischen Laden des externen Assembler-Unterprogramms SPACE als COM-Datei SPACE.OVX gezeigt. Da die Parameterübergabe von SPACE.OVX im Doppelregister HL erwartet wird – Konvention bei Aufruf von Basic-80 oder dBase II aus – ist ein Aufrufvermittler als PROCEDURE FREE in Form von Assembler-INLINE-Code zur Parameteranpassung dazwischengeschaltet.
Mit dem Unterprogramm SPACE.OVX werden je nach Parameteraufbau entweder der noch freie Platz oder die noch freien DIRectory-Entries des ausgewählten Laufwerkes ermittelt (siehe auch /3/ und /4/).
Wie die hier gezeigte Methodik der Anwendung externer Unterprogramme in Turbo-Pascal mit dem Run-Time-Lader XPOLADER.INC zum Zwecke der Nutzung vorhandener Software, beispielsweise mit Fortran, eingesetzt werden kann, soll anhand der Adaption des Beitrages aus /5/ für ein Turbo-Pascal-Programm gezeigt werden, womit auch Fortran-Experten ein Einstieg in Turbo-Pascal schmackhaft gemacht werden soll. Im Bild 3 ist das Programm GRAFIDRU.PAS gezeigt, mit dem x,y-Werte-Tupel ($0 < x \leqq 10.84$ bzw. $0 < y \leqq 9.03$) aus einer ASCII-Datei im TPA als Pixel-Bild in 16384 Byte aufgebaut und danach mit dem externen Unterprogramm DRUCK.OVX – gewonnen aus den Fortran-Subroutinen DRUCK, DRZEIL sowie den Assembler-Unterprogrammen SRL und ANDB aus /5/ – auf einem Nadeldrucker als Bild im Format 10,84 cm × 9,03 cm (Breite mal Höhe) ausgedruckt werden.
In Bild 4 ist ein Anwendungsbeispiel demonstriert. Das vorgelagerte Programm GRAFIK

Fortsetzung auf Seite 211

```
PROCEDURE XPOLADER (VAR DSN:Str11;VAR LAD,EAD:INTEGER);
{DSN::= Dateiname als "NNNNNNNNTTT" der Datei (z.B. TTT:=OVX )
        auf akt. Laufwerk, die [als externes Unterprogramm]
        auf die Ladeadresse ( enthalten in LAD ) dynamisch
        geladen werden soll.                                  }
{LAD::= enthaelt die Ladeadresse des TPA-Bereiches zur
        Aufnahme d. Datei [d. externen Unterprogramms] aus DSN.
        Rueckgabe: LAD unveraendert OR  Ø --> DSN not found. }
{EAD::= enthaelt eine obere maximale zum Laden des
        (<anz>+1)-ten Sektors nicht mehr nutzbare TPA-Adresse.
        EAD >= LAD+(<anz>+1)*128;
        oder bis BDOS-Beginn  ===>  EAD:=$FFFF;
        Rueckgabe: EAD unveraendert OR  Ø --> Datei zu gross.}
{   Eine Assemblerroutine SPACE.MAC als externes Unterprogramm
    soll z.B. auf die Ladeadresse ØA4ØFh gelegt werden:
B>A:M8Ø SPACE,TTY:=SPACE/z/x/c/m<CR>
B>A:L8Ø /P:A4ØF,SPACE/E<CR>
B>A:POWER SAVE SPACE.OVX  A4ØF 5<CR>                          }
{  Damit gilt:                                                }
{  TYPE Str11=STRING[11];                                     }
{  VAR DSN:Str11; LAD,EAD:Integer;                            }
{  begin DSN:='SPACE   OVX';  LAD:=$A4ØF;
          EAD:=LAD+(5+1)*128;  oder:  EAD:=$FFFF;
          XPOLADER(DSN,LAD,EAD); IF (LAD=Ø) OR (EAD=Ø) THEN HALT
   end;                                                       }
BEGIN  {Modul XPOLADE.INL}
INLINE (
$2A/DSN     /$7E/$FE/$ØB/$2Ø/$76/$23/$11/*+131  /$Ø1/$ØB/$ØØ/$ED/
$BØ/$3E/$ØØ/$32/*+132  /$32/*+149  /$ØE/$19/$CD/$Ø5/$ØØ/$3C/$32/
*+1Ø8  /$11/*+1Ø5  /$ØE/$ØF/$CD/$Ø5/$ØØ/$FE/$FF/$28/$5Ø/$DD/$2A/
LAD    /$DD/$5E/$ØØ/$DD/$56/$Ø1/$D5/$D1/$D5/$ØE/$1A/$CD/$Ø5/$ØØ/
$11/*+75    /$ØE/$14/$CD/$Ø5/$ØØ/$E1/$A7/$2Ø/$64/$Ø1/$8Ø/$ØØ/$Ø9/
$E5/$D1/$E5/$FD/$2A/EAD     /$FD/$6E/$ØØ/$FD/$66/$Ø1/$B7/$ED/$42/
$B7/$ED/$52/$E1/$3Ø/$ØA/$FD/$36/$ØØ/$ØØ/$FD/$36/$Ø1/$ØØ/$18/$4Ø/
$ØB/$Ø9/$ED/$4B/$Ø6/$ØØ/$B7/$ED/$42/$3Ø/$EB/$D5/$18/$BB/$DD/$2A/
LAD    /$DD/$36/$ØØ/$ØØ/$DD/$36/$Ø1/$ØØ/$18/$24/$ØØ/$ØØ/$ØØ/$ØØ/
$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/
$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ)
END;   {Modul XPOLADE.INL}
```

***Bild 1** Run-Time-Lader PROCEDURE für Turbo-Pascal*

```
Ø PROGRAM SPACETST;
Ø (* Demonstrationsprogramm fuer den Einsatz des RUN-Time-Laders
Ø XPOLADE zum dynamischen Nachladen der externen Assembler-
Ø Routine SPACE.OVX zur Bestimmung des noch verfuegbaren
Ø Speicherplatzes bzw. der noch freien DIRectory-Entries auf
Ø der Diskette des im Parameter mitgeteilten Laufwerkes.        *)
Ø (* Option: <<O>>   Bei Eingabe von <<C>> noch <<E>> auf den
Ø                    Wert A4ØE bzw. A3FF setzen.                *)
Ø TYPE Str11=STRING[11]; Str4=STRING[4];
Ø VAR  DSN:Str11; LAD,EAD:INTEGER; Mparm:Str4;   B:CHAR;
Ø
Ø (*$I XPOLADER.INC       <=====     RUN-Time-Lader   =====     *)
Ø
Ø PROCEDURE ReadTRM (VAR Key:CHAR);   (* Lesen vom Geraet TRM *)
Ø BEGIN Read(KBD,Key); IF Key>=' ' THEN Write(Key) END;
Ø
Ø (* FREEK dient zum Aufrufen des externen Unterprogramms
Ø SPACE.OVX auf ØA4ØFh wegen der hier benutzten Parameter-
Ø uebergabekonvention (Reg. hl zeigt auf Laengenbyte von Xparm *)
Ø PROCEDURE FREEK (VAR Xparm:Str4); BEGIN
1 INLINE(           (*          .Z8Ø                            *)
1 $2A/XPARM         (*          LD HL,(XPARM)                   *)
1 /$ED/$73/*+$ØØ1E  (*          LD (RETTE),SP                   *)
1 /$31/*+$ØØ1B      (*          LD SP,RETTE                     *)
1 /$CD/$ØF/$A4      (*          CALL ØA4ØFH                     *)
1 /$ED/$7B/*+$ØØ14  (*          LD SP,(RETTE)                   *)
1 /$18/$12          (*          JR RETTE+2                      *)
1 /$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ/$ØØ
1                   (*          DEFS 16                         *)
1 /$ØØ/$ØØ/$ØØ)
1                   (*RETTE     DEFS 2                          *)
1                   (*          END                             *)
1                               END;
Ø BEGIN  (*         Dynamische Nachladen von SPACE.OVX          *)
1   DSN:='SPACE   OVX'; LAD:=$A4ØF;EAD:=LAD+(5+1)*128;EAD:=$FFFF;
1   XPOLADER(DSN,LAD,EAD);
1   IF LAD=Ø THEN Writeln('Datei ',DSN,' nicht gefunden!');
1   (*SPACE.OVX  koennte  vorab mit EXEC/POWER/L8Ø geladen sein*)
1   IF EAD=Ø THEN BEGIN Writeln('Datei zu gross --> CANCEL');
2          HALT END;
1 REPEAT            (* *** Hauptzyklus *** *)
2   Write(#13#1Ø+'======== GO ON ===========>'+#13#1Ø#1Ø,
2                 'ENTER Laufwerk (a..p) fuer FREE-Service:');
2   ReadTRM(B);           B:=UpCase(B);                    Writeln;
2   Mparm:=B+'DIR';    FREEK(Mparm);                       Writeln;
2   Write('Anzahl freier  DIRECTORY-Entries  auf  Lw.   ',B,
2                                       ' : '+Mparm+#13#1Ø#1Ø#1Ø);
2   Mparm:=B+'   ';    FREEK(Mparm);
2   Writeln('F r e i e   K a p a z i t a e t   auf  Lw.  '+
2   CHR(134),B,CHR(132)+': '+Mparm+CHR(134)+' KByte'+CHR(132));
2   Writeln(#13#1Ø#1Ø);  Write('Programm beenden    (j/N) ? ');
2   ReadTRM(B);  Writeln;                      B:=UpCase(B)
2 UNTIL B='J'                                               END.
```

***Bild 2** Dynamisches Laden des externen Unterprogramms SPACE.OVX mittels Run-Time-Lader XPOLADER*

Einführung in Forth-83

Teil 3

Dr. Hartmut Pfüller (Leiter),
Dr. Wolfgang Drewelow, Dr. Bernhard Lampe, Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

3. Das Forth-Betriebssystem

Forth wird vielfach nicht nur als Programmiersprache, sondern als Betriebssystem, als Programmierwerkzeug oder als Softwareumgebung bezeichnet. Ein Grund dafür ist sicherlich, daß die Programmierung nicht wie bei den meisten Sprachen üblich in der Reihenfolge Editor, Compiler, Debugger mit der immer dazwischenstehenden Stufe Betriebssystem abläuft, sondern einheitlich in der Forthumgebung erfolgt (eventuell auch unter Nutzung von äquivalenten Forthkomponenten). Ein anderer Grund liegt darin, daß Forth bereits im Kern eine Schichtenstruktur besitzt, die stark an den Aufbau einfacher Betriebssysteme angelehnt ist: In der untersten Schicht realisiert Forth auf einem sehr niedrigen Niveau den Bezug zur Standardperipherie (Tastatur, Bildschirm, Massenspeicher). In der darüberliegenden Schicht arbeiten dann schon geräteunabhängige Worte zur Zeichenkettenverarbeitung und -konvertierung sowie Funktionen des virtuellen Massenspeicherkonzepts. In einer weiteren Schicht erfolgt die Definition des Kommandointerpreters des Systems.
Die Schnittstelle zwischen dem Forthsystem und der Peripherie kann sehr schmal gehalten werden und ermöglicht deshalb eine schnelle Implementierung von Forth auch auf der nackten Hardware. Forth kann damit auch als kompaktes und sehr flexibles Betriebssystem auf beliebigen, aufgabenspezifisch zugeschnittenen Rechnerkonfigurationen eingesetzt werden und bietet darüber hinaus (im Unterschied zu anderen kleineren Betriebssystemen) den gesamten eigenen Sprachumfang für die Programmentwicklung.

3.1. Konsolfunktionen

3.1.1. Zeichenweise Ein- und Ausgaben

Die beiden grundlegenden Worte für die Konsolbedienung sind die Einzelzeichentreiber KEY und EMIT. Die Funktion **KEY** wartet auf das nächste über die Tastatur eingegebene Zeichen und legt es als 16-Bit-Wert auf den Datenstapel. In den unteren 7 Bits befindet sich die ASCII-Information. Darüber hinaus werden aber alle Bits entgegengenommen – die höherwertigen Bits können systembedingt auch ungleich Null sein. Die zu KEY komplementäre Aktion **EMIT** gibt das in den niederwertigen 7 Bits befindliche ASCII-Zeichen einer auf dem Stapel liegenden Zahl an das Terminalgerät aus. Die höherwertigen Bits können wieder gerätespezifisch angewandt werden. Mit Hilfe von EMIT ist die Definition von weiteren zum Kernwortschatz gehörenden Konsolfunktionen möglich. Das Wort **SPACE** zur Ausgabe eines Leerzeichens (ASCII-Code 32) könnte im Forthsystem so definiert sein:

```
32  CONSTANT  BL

:  SPACE      ( ===> )
   BL  EMIT  ;
```

Die Ausgabe von mehreren ASCII-Leerzeichen kann durch die Verwendung von SPACES erfolgen, wobei auf dem Stapel die Anzahl erwartet wird. Eine Definition von SPACES wäre unter Verwendung von SPACE innerhalb einer **DO-LOOP**-Konstruktion möglich:

```
:  SPACES      ( +n ===> )
   0  MAX  ?DUP  IF  0  DO  SPACE
                           LOOP
               THEN  ;

:  CR          ( ===> )

   13  EMIT  10  EMIT  ;
```

Das Wort **CR** führt einen Wagenrücklauf und Zeilenvorschub (Carriage Return – Linefeed, ASCII-Codes 13 und 10) aus.
Eine häufig implementierte, allerdings nicht im Standard fixierte Funktion ist **KEY?**. Dieses Wort testet den aktuellen Status des Eingabegerätes: Liegt eine Tastenbetätigung vor, so wird ein True-Flag auf den Stapel übergeben, andernfalls ein False-Flag.

3.1.2. Zeichenkettenausgabe

Eine Forthzeichenkette (Forthstring) umfaßt in ihrer Standarddarstellung zwei Informationsanteile: die Längeninformation und die eigentliche Zeichenfolge. Die Längeninformation befindet sich im ersten Byte (Zählbyte). Danach schließt sich die Zeichenfolge an:

```
100 101 102 103 ...       ---->    aufsteigende Adreßrichtung

!   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !   !
!18 ! D ! a ! s !   ! i ! s ! t !   ! e ! i ! n !   ! S ! t ! r ! i ! n ! g !
!___!___!___!___!___!___!___!___!___!___!___!___!___!___!___!___!___!___!___!
  ^
  !                           ASCII-Zeichen
Zählbyte
(englisch: count)
```

Erfordert ein Programm eine getrennte Behandlung der Längeninformation und der Zeichenfolge, so kann das Wort **COUNT** benutzt werden. COUNT erwartet auf dem Stapel die Anfangsadresse einer Zeichenkette und erzeugt daraus zwei Ergebnisparameter: Auf der Stapelspitze liegt der Inhalt des Zählbytes, darunter die Adresse des ersten Zeichens der Zeichenkette. Das Wort **TYPE** baut auf den beiden von COUNT gelieferten Parametern auf und bringt die Zeichenkette zur Anzeige. COUNT und TYPE werden deshalb häufig im Zusammenhang benutzt:

```
100  COUNT  TYPE (cr)_Das_ist_ein_String_ok
```

Das Wort **-TRAILING** geht von den gleichen Ausgangsparametern wie TYPE aus. Es prüft, ob das Ende der Zeichenkette aus Leerzeichen besteht, die bei der Weiterverarbeitung der Kette weggelassen werden können. Die auf dem Stapel liegende Adresse des ersten Zeichens bleibt unverändert, nur das Zählbyte wird gegebenenfalls verkleinert. Die Befehlsfolge
addr COUNT -TRAILING TYPE
sorgt also gegenüber der Folge
addr COUNT TYPE
für eine Verkürzung der Ausgabeaktion, sofern am Stringende Leerzeichen standen. Soll innerhalb eines Forthprogramms die Ausgabe einer festen Zeichenkette erfolgen, so kann dazu das Wort ." benutzt werden. Die Zeichenfolge **ccc** der Sequenz ." **ccc**" wird innerhalb einer Wortdefinition in eine Forthzeichenkette umgewandelt und bei der späteren Ausführung dieses Wortes ausgegeben (die Zeichen zwischen den Anführungsstrichen, aber ohne das dem Wort ." folgende Leerzeichen):

```
:  TREFFER     ( n ===> )
   ." Das waren "  .   ." Treffer ! "  ;

5  TREFFER(cr)_Das_waren_5_Treffer_!_ok
```

3.1.3. Kettenorientierte Eingabe

Ein zentrales Wort für die Zeicheneingabe ist **EXPECT**. Es wird insbesondere auch vom Textinterpreter benutzt. Eingangsparameter für EXPECT sind die Adresse, auf der die geholte Zeichenkette im Speicher abgelegt werden soll (Adresse des ersten Zeichens, nicht Adresse der ersten Zählbytes!), und eine Längenvorgabe. Die Zeichenketteneingabe wird durch **(cr)** oder durch das Erreichen der Längenvorgabe beendet. Durch EXPECT wird das Zählbyte der Zeichenkette nicht gesetzt. Dafür steht die Anzahl der eingegebenen Zeichen in der Systemvariablen **SPAN**. Nachfolgend ist ein Beispiel für die Anwendung von EXPECT und SPAN zur Umwandlung einer über die Tastatur eingegebenen Folge von Zeichen (maximal 20) in eine Forthzeichenkette im Standardformat angegeben.

```
:  GET_STRING     ( addr ===> )
   DUP  1+  20  EXPECT
   SPAN  @  SWAP  C!  ;
```

Das Wort **GET_STRING** erwartet dabei als Eingangsparameter die Zieladresse des Strings. Nachdem mit EXPECT die Zeicheninformation auf die Zieladresse + 1 abgelegt wurde, wird das Zählbyte unter Nutzung der in SPAN stehenden Längeninformation gesetzt. Die Benutzung von GET_STRING zeigt folgendes Beispiel:

```
VARIABLE  KETTE       19  ALLOT(cr)_ok
KETTE  GET_STRING  (cr)_Hallo, Paul !  ok
KETTE  COUNT  TYPE  (cr)_Hallo,_Paul_!_ok
```

Der Ausdruck **19 ALLOT** nach der Variablendefinition erweitert den Speicherplatz für die Variable um 19 zusätzliche Bytes.

3.2. Zahlenkonvertierung

Im Forthsystem sind Funktionen zur Ein- und Ausgabekonvertierung von Ganzzahlen implementiert. Forth bietet die einzelnen Bestandteile dieser Funktionen auch nach außen an und erlaubt damit dem Anwender, eigene, modifizierte Konvertierungen zu entwerfen.

3.2.1. Steuern der Zahlenbasis

Zentraler Bezugspunkt für die Ein- und Ausgabekonvertierung ist die aktuelle Zahlenbasis. Der Wert dieser aktuellen Zahlenbasis ist in der Systemvariablen **BASE** gespeichert. Eine Veränderung von BASE wirkt damit unmittelbar auf die Konvertierung. Die dezimale Zahlenbasis wird durch **DECIMAL** eingestellt, was intern wie folgt definiert sein könnte:

```
:  DECIMAL      ( ===> )
   10  BASE  !  ;
```

Analog können auch andere Worte zum Einstellen der Zahlenbasis vereinbart werden:

```
DECIMAL

 :  HEX          ( ===> )
    16  BASE  !  ;
 :  OCTAL        ( ===> )
    8  BASE  !  ;
 :  BINARY       ( ===> )
    2  BASE  !  ;
```

Die Nutzung dieser Möglichkeit soll hier anhand der Erzeugung einer Tabelle für unterschiedliche Zahlendarstellungen demonstriert werden. Das Wort **TABELLE** stellt in jedem Schleifendurchlauf einen Zählwert als Dezimal-, Oktal-, Hexadezimal- und Binärzahl dar:

```
:  TABELLE      ( n ===> )
   BASE  @  SWAP
   1  DO  CR  HEX       I  .
              DECIMAL   I  .
              OCTAL     I  .
              BINARY    I  .
     LOOP
   BASE  !  ;
```

Die Tabellenlänge wird dabei als Parameter auf dem Stapel übergeben. Die Rahmenaktionen **BASE @ SWAP** und **BASE !** sorgen dafür, daß zunächst der Wert der Zahlenbasis auf dem Stapel abgelegt wird und daß zum Abschluß dieser noch immer auf dem Stapel befindliche Wert wieder in die Variable BASE eingetragen wird. Ohne diesen Rahmen wäre nach der Abarbeitung von TABELLE immer die zuletzt benutzte binäre Zahlenbasis eingestellt.

3.2.2. Ausgabekonvertierung

Die Ausgabekonvertierung arbeitet nach folgendem Prinzip: Zunächst wird die auf dem Stapel liegende Zahl gegebenenfalls zu einer 32-Bit-Zahl ergänzt. Danach wird wiederholt das Wort # aufgerufen. Damit wird die Zahl jedesmal durch die aktuelle Zahlenbasis dividiert, bis der Quotient 0 ist. Die sich jeweils ergebenden Reste (die zwischen 0 und dem Wert der Zahlenbasis minus 1 liegen) werden in die entsprechenden ASCII-Ziffernzeichen umgewandelt und in der Reihenfolge von der kleinsten bis zur höchsten Stelle (von rechts nach links) in einem Stringzwischenspeicher abgelegt. Nach Beendigung der eigentlichen Wandlung kann der fertige String ausgegeben werden. Der Aufbau des Zahlenstrings wird durch die Operation <# initialisiert. Die eigentliche Konvertierung darf erst nach Abarbeitung dieser Funktion beginnen. Die Sammeloperation **#S** führt die Funktion # gleich so oft nacheinander aus, bis der 32-Bit-Quotient 0 ist, also bis die gesamte Zahl zu Ende gewandelt ist. Die doppelt genaue Null (letzter Quotient) bleibt nach der Operation **#S** noch auf dem Stapel liegen, damit die gleichen Stapelbedingungen wie bei der Anwendung von # gesichert werden. Der Abschluß der numerischen Ausgabekonvertierung erfolgt durch #>, wobei auf dem Stapel der letzte Quotient erwartet wird. Ausgangsparameter sind die Adresse des ersten Stringzwischenspeichers und das Zählbyte auf der Stapelspitze. Damit kann unmittelbar nach #>das Wort TYPE angewendet werden. Die interne Definition der Funktion **U.** zur Ausgabe einer vorzeichenlosen ganzen Zahl könnte damit so aussehen:

```
:  U.        ( n ===> )
   0         ( Ergaenzung auf 32 Bit)
   <#  #S  #>  TYPE  SPACE  ;
```

Möchte man zur Strukturierung der Ausgabe an einer bestimmten Stelle in der Ergebnisdarstellung zusätzlich ein ASCII-Zeichen (zum Beispiel einen Dezimalpunkt) unterbringen, so kann dies durch Ausführung der Funktion **HOLD** an der entsprechenden Stelle des Konvertierungsvorgangs erfolgen. HOLD erwartet als Eingangsparameter den ASCII-Code des einzufügenden Zeichens und hinterläßt keinen Ausgangsparameter. Eine strukturierte Ausgabe unter Verwendung von HOLD kann beispielsweise bei der Anzeige der Uhrzeit benutzt werden. Soll für eine auf dem Stapel liegende Sekundenzahl eine Zeitanzeige in der Form Std:Min:Sek erfolgen, so ist das unter Benutzung der Definitionen möglich:

```
:  SEXTAL      ( ===> )
   6  BASE  !  ;

:  :XX         ( n1 ===> n2)
   DECIMAL  #  SEXTAL  #  58  HOLD  ;

:  .ZEIT       ( n ===> )
   BASE  @  SWAP
   0  <#  :XX  :XX  DECIMAL  #S  #>
   TYPE  SPACE  BASE  !  ;


7572  .ZEIT  (cr) 2:06:12 ok
```

Das Wort :**XX** dient der Darstellung der Sekunden bzw. Minuten, die jeweils im Bereich von 0 bis 59 liegen. Hier erfolgt zunächst die Konvertierung der rechten Stelle, die zwischen 0 und 9 liegt (dezimale Basis), dann die Konvertierung der linken Stelle (Bereich 0 bis 5, sextale Basis). Abschließend wird „links" mit **58 HOLD** ein Doppelpunkt erzeugt. In **.ZEIT** wird zunächst die Ergänzung auf 32 Bit vorgenommen. Danach erfolgt die Ausgabe der Sekunden und Minuten durch das Wort :**XX** und mit **#S** die Wandlung des Restes, das heißt der Stunden. Abschließend wird der erzeugte String mit TYPE ausgegeben.

Das Wort **SIGN** dient der Vorzeichenbehandlung bei der Ausgabekonvertierung. Es fügt den ASCII-Code eines Minuszeichens in den Zahlenstring ein, falls der auf dem Stapel liegende Wert negativ ist. SIGN wird zumeist direkt vor #> angewendet (das heißt, am Ende des Konvertierungsvorgangs). Es plaziert damit gegebenenfalls ein Minuszeichen an die am weitesten links stehende Stelle des Zahlenstrings (vor die Ziffern). Bei Konvertierungsaktionen wird als Zeichenkettenzwischenspeicher vielfach ein dem Anwender zugänglicher Speicher benutzt, dessen Startadresse durch das Wort **PAD** geliefert wird. Allerdings sind Forthworte, die unter Benutzung von PAD geschrieben wurden, im allgemeinen nicht wiedereintrittsfähig.

Mit den oben beschriebenen Teilfunktionen zur Ausgabekonvertierung lassen sich (wie am Beispiel von U. beschrieben) sämtliche in Forth vorhandenen Worte zur Zahlenausgabe definieren. In vielen Implementierungen von Forth-83 sind die Worte **.R** und **D.R** vorhanden. Diese Funktionen nehmen die rechtsbündige Ausgabe einer einfach- bzw. doppeltgenauen Integerzahl vor, wobei eine auf der Stapelspitze stehende positive Zahl die Weite des Ausgabefeldes festlegt.

3.2.3. Eingabekonvertierung

Die vom Textinterpreter benutzte zentrale Funktion zur Eingabekonvertierung ist

CONVERT (+d1 addr1 => +d2 addr2)

Von der ab adr1+1 stehenden Zeichenkette wird das erste ASCII-Zeichen geholt. Sofern dieses Zeichen ein gültiges Ziffernsymbol entsprechend der eingestellten Zahlenbasis ist, wird die konvertierte Ziffer zum doppelt genauen Produkt aus Zahlenbasis und 32-Bit-Stapelwert addiert. Die Adresse innerhalb des Zahlenstrings wird erhöht. Dieses Verfahren wird solange fortgesetzt, bis das nächste anliegende ASCII-Zeichen nicht mehr als Ziffer interpretierbar ist. Die Adresse addr2 dieses nicht konvertierbaren Zeichens im String verbleibt ebenso wie das doppelt genaue Konvertierungsergebnis +d2 auf dem Stapel.

3.3. Massenspeicheranbindung

3.3.1. Virtuelle Speichertechnik

Forth bietet dem Anwender zur Verwaltung des Massenspeichers ein sehr einfaches, aber leistungsfähiges Konzept an. Der gesamte Massenspeicher (zum Beispiel Band, Diskette oder Festplatte) ist in aufeinanderfolgende Blöcke von jeweils 1024 Byte eingeteilt. Der Arbeit mit den Massenspeicherdaten eines Blockes erfolgt nun aus der Sicht des Forthnutzers nicht über direkte Lese- oder Schreibzugriffe zum Externspeicher, sondern nach Angabe der Nummer des zu bearbeitenden Blockes über normale Speicherzugriffe zu einem reservierten Hauptspeicherbereich, dem sogenannten Block-Puffer-Bereich. Der eigentliche Massen-

speicherzugriff erfolgt verdeckt innerhalb derjenigen Befehle, die Forth zur Umwandlung der Massenspeicheradresse (Blocknummer) in die tatsächliche Hauptspeicheradresse zur Verfügung stellt (virtuelles Speicherprinzip).

3.3.2. Verwalten der Datenblöcke

Das Wort **BLOCK (u => addr)** liefert die Anfangsadresse eines 1024 Byte langen Puffers, in dem der Inhalt des Blocks mit der Blocknummer u steht. BLOCK arbeitet intern folgende Operationen ab: Falls der Block u noch nicht in einem der Puffer verfügbar ist (andernfalls ist keine weitere Bearbeitung erforderlich), erfolgt die Zuweisung eines Puffers. Steht ein als geändert gekennzeichneter Block in diesem Puffer, wird dieser vor der Neuvergabe des Puffers zurückgeschrieben. Dann wird der Block mit der Nummer u vom Massenspeicher in den Puffer transferiert. Die Funktion **BUFFER (u => addr)** ist inhaltlich identisch mit der Funktion BLOCK bis auf den Umstand, daß die Blockleseaktion nicht ausgeführt wird. Der Pufferinhalt ab addr ist damit unbestimmt. Sinnvoll läßt sich dieses Wort anwenden, wenn ein Block vollständig neu beschrieben werden soll; eine vorherige Leseaktion wäre in diesem Fall unnötig. Es gilt die Konvention, daß nur der zuletzt mit BLOCK oder BUFFER angesprochene Pufferinhalt gültig ist. Mit dem Wort **UPDATE** kann dem Forthsystem mitgeteilt werden, daß der Pufferinhalt des aktuell in Bearbeitung befindlichen Blocks aktualisiert worden ist. So markierte Blöcke werden später automatisch auf den Massenspeicher geschrieben, wenn der Puffer für das Speichern eines anderen Blocks benötigt wird. Bei Beenden der Arbeit mit dem Massenspeicher oder auch nur zu Sicherungszwecken ist es sinnvoll, alle als aktualisiert gekennzeichneten Blöcke auf den Massenspeicher zu transferieren, ohne daß vorher Leseaktionen anderer Blöcke erfolgen müssen. Eine solche Funktion erfüllt **SAVE-BUFFERS**. Dieses Wort läßt die Pufferinhalte unverändert und setzt die Aktualisierungskennung zurück. Eine ähnliche Reaktion wird durch das Wort **FLUSH** ausgelöst. Auch hier werden die aktualisierten Blöcke sofort auf den Massenspeicher geschrieben. Im Gegensatz zu **SAVE-BUFFERS** werden aber sämtliche Blockpuffer freigegeben, so daß bei einem neuen Zugriff auf einen Block eine Leseoperation auf dem Massenspeichermedium notwendig wird. Sinnvoll ist diese Funktion zum Beispiel anwendbar, wenn ein Wechsel des Massenspeichermediums vorgesehen ist. So wird vermieden, daß die noch im Puffer befindlichen Blöcke des alten Mediums fälschlicherweise auch als Blöcke des neuen Mediums interpretiert werden. Der Veranschaulichung der Arbeit mit dem Massenspeicher soll ein einfaches Beispiel dienen: Es ist das externe Abspeichern von 16-Bit-Daten zu organisieren:

```
100    CONSTANT  BLOCK0
1024   CONSTANT  B/BUF

:  POSITION
   ( messwertnr ===> pufferoffset blocknr)
   2  *  B/BUF  /MOD  SWAP  BLOCK0  +  ;

:  !WERT          ( wert messwertnr ===> )
   POSITION  BLOCK  +  UPDATE  !  ;
```

Hier markiert **BLOCK0** die Nummer eines Massenspeicherblocks, von dem ab die Aufzeichnung der 16-Bit-Daten erfolgen soll. **B/BUF** ist eine Konstante, die die Länge eines Blocks bzw. eines Blockpuffers angibt. Durch **POSITION** wird unter Benutzung von BLOCK0 und B/BUF das Umrechnen der Datenwertnummer (virtuelle Adresse) in die Massenspeicheradresse (Blocknummer, Position innerhalb des Blocks) vorgenommen. Die Operation **!WERT** zum Abspeichern eines 16-Bit-Datenwertes benutzt wiederum zunächst POSITION, macht den entsprechenden Massenspeicherblock mit BLOCK verfügbar, markiert mit UPDATE den Block als geändert und nimmt dann die eigentliche Änderung durch das Eintragen des Meßwertes auf die errechnete Pufferposition vor. Die ab BLOCK0 abgespeicherten Meßdaten können mit dem nachfolgend definierten Zugriffsbefehl wieder gelesen werden:

```
:  @WERT          ( messwertnr  ===>  wert)
   POSITION  BLOCK  +  @  ;
```

Die Operationen **!WERT** und @**WERT** arbeiten in weitgehender Analogie zu den normalen Speicherzugriffsbefehlen ! und @. Allerdings sind die bei !WERT und @WERT benutzten Meßwertnummern die Adressen eines „echten“ 16-Bit-Speichers, das heißt, zwei aufeinanderfolgende Adressen adressieren zwei sich nicht überlappende 16-Bit-Speicherzellen. Nachfolgend ein Bedienungsbeispiel:

```
10 0 !WERT 15 1 !WERT 30 2 !WERT (cr)_ok
0 @WERT . 1 @WERT . 2 @WERT . (cr)_10_15_30_ok
```

Das vorgestellte Massenspeicherkonzept ist allen Systemen gemeinsam, die dem Standard entsprechen. Allerdings lassen sich durch unterschiedliche Implementierungen der Lese- und Schreiboperationen, die in BLOCK und BUFFER benutzt werden, sehr verschiedene Forthsysteme erzeugen, ohne daß dabei eine Verletzung des Standards vorliegen würde. So ist es in Standalone-Versionen üblich, selbstdefinierte physische Gerätetreiber in die Schreib- und Leseoperationen einzusetzen, wobei keine Kompatibilität mit anderen Datenformaten erforderlich ist. Werden dagegen die Befehle BUFFER und BLOCK bei vorhandenem Hostbetriebssystem auf die BIOS-Ebene aufgesetzt, so wird die Kompatibilität des Forthsystems zum Hostbetriebssystem auf der Ebene des physischen Datenformats erreicht. In beiden Fällen stellt sich der gesamte Massenspeicher aus der Sicht des Forthprogrammierers als eine homogene Einheit dar, die nur durch die Blockeinteilung strukturiert ist. Vielfach benutzen Forthsysteme die Lese- und Schreiboperationen eines Hostbetriebssystems auf der DOS-Ebene. Zum einen wird hierdurch eine Dateistrukturierung der Forthprogrammquellen und der Daten möglich; diese brauchen nun nicht mehr nur durch die Blocknummer adressiert werden. Eine Handhabung im klassischen Sinn bleibt weiterhin möglich, indem immer nur über ein- und derselben Datei gearbeitet wird. Zum anderen können die vom Betriebssystem gebotenen Systemhilfen und Standardprogramme genutzt werden. Außerdem kann der Vertrieb von Forthsystemen durch die Orientierung auf Standardbetriebssysteme vereinfacht werden. Systeme mit DOS-Anbindung stellen dem Programmierer üblicherweise Worte zur Dateibehandlung zur Verfügung, die sich weitgehend an den vom Betriebssystem angebotenen Funktionen orientieren.

3.3.3. Speichern von Quelltext

Größere Quellprogramme wird man vor der Kompilation unter Verwendung eines Editors erzeugen und auf dem Massenspeicher konservieren. In Forth wird solcher Quelltext gewöhnlich in Blöcke zu jeweils 1024 Byte gegliedert, wobei ein Block 16 Zeilen mit je 64 Zeichen beinhaltet:

```
  Datei A:FORTH3.CF2   Block 1
\ MP3; Ausgaben                          WD 29-Nov-88

32  CONSTANT  BL

:   SPACE     ( ===> )
    BL  EMIT  ;

:   SPACES    ( +n ===> )
    0  MAX  ?DUP  IF  0  DO  SPACE
                             LOOP
                      THEN  ;

:   TREFFER   ( n ===> )
    ." Das waren "  .  ." Treffer ! "  ;
```

Für das Erzeugen eines solchen Quellblocks finden sowohl in Forth geschriebene Zeileneditoren als auch komfortable Sreeneditoren Anwendung. Ihre Beschreibung ist im Rahmen des Kurses nicht möglich; Informationen darüber sollten der Dokumentation des entsprechenden Forthsystems entnommen werden.

Ein fertiger Block kann mit dem Befehl **LOAD** interpretiert werden, wobei die Blocknummer auf der Stapelspitze erwartet wird. Dabei gilt für den Blockquelltext die gleiche Syntax, die auch bei der interaktiven Kommandoausgabe gültig ist. So kann ein Quelltext sowohl auszuführende Kommandos als auch neue Definitionen enthalten. Diese Identität bei der Interpretation von Quellblöcken und von interaktiven Eingaben rührt daher, daß durch den Befehl LOAD nur ein zeitweiliges Umlenken des Eingabedatenstroms vorgenommen wird. Die Anzeige des Inhalts eines Quellblocks kann auch ohne einen Editor mit dem Befehl **LIST** erfolgen. Umfaßt der einzugebende Programmquelltext mehrere Blöcke, so können diese durch mehrfache Anwendung der Funktion LOAD geladen werden. Eine andere Möglichkeit besteht darin, Blöcke mit dem Wort --> zu verketten. Das Wort --> steht auf einem Quellblock nach dem letzten Quelltextwort und sorgt dafür, daß auch der nachfolgende Block geladen wird. Das Einfügen von Kommentaren in den Quelltext ist unter Verwendung von runden Klammern, das heißt der Wortkombination (und) möglich. Der Text, der zwischen diesen Zeichen steht, wird vom Interpreter ignoriert. Die öffnende Klammer ist ein normales Forthwort, hinter dem (natürlich) ein Leerzeichen stehen muß! Die schließende Klammer wird analog dem " bei ." als Begrenzerzeichen

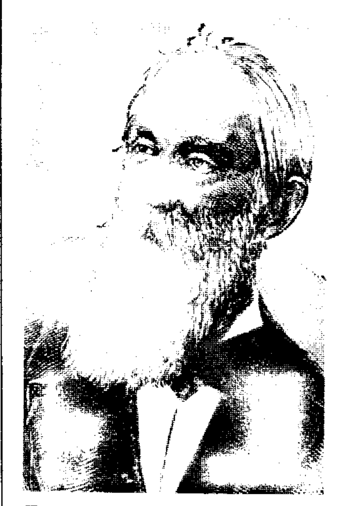
verwendet. Ähnlich kann die Kombination .(und) benutzt werden. Sie gibt die zwischen den Klammern stehende Zeichenkette auf dem Terminal aus. Diese Funktion kann zum Beispiel benutzt werden, um beim Laden von langen Quelltextpassagen Zwischeninformationen anzuzeigen.
Forth läßt dem Programmierer viele Freiheiten bei der Formulierung eines Programms. Damit ist die Qualität eines Forthprogramms empfindlich gegenüber der Fähigkeit des Programmierers, diese Freiheiten sinnvoll ausnutzen zu können. Neben solchen Faktoren wie Festlegung der Wortschnittstellen (Dekomposition) und der Namensgebung ist die Gestaltung des Quelltextes von großer Bedeutung für die Übersichtlichkeit, die Lesbarkeit und die Möglichkeiten zur Wartung eines Forthprogramms. An dieser Stelle sollen deshalb einige ausgewählte Empfehlungen für das Layout von Quelltexten genannt werden (siehe auch /2/ in Teil 1):
– Reservieren der Zeile 0 eines jeden Blocks für einen Kommentar, in dem der Zweck des Blocks und der „Stempel" (Initialen des Programmierers und Datum der letzten Revision) enthalten sind
– Benutzen von Leerschritten und Einrükkungen zur Verbesserung der Lesbarkeit
– Notation des Stapeleffekts für jede Doppelpunkt- oder Codedefinition, die Stapelwerte benutzt oder hinterläßt
– Beginn aller Definitionen am linken Zeilenrand, immer nur eine Definition pro Zeile.

wird fortgesetzt

Tafel 3 Kurzbeschreibung der Forthworte

Name	Stapeleffekt	Beschreibung
Konsolfunktionen		
EMIT	(c ===>)	Ausgabe eines Zeichens
KEY	(===> c)	Zeichen einlesen
KEY?	(===> ?)	wahr für Tastatur ist betätigt
CR	(===>)	neue Zeile
SPACE	(===>)	Ausgabe eines Leerzeichens
SPACES	(+n ===>)	Ausgabe von +n Leerzeichen
Zeichenkettenbefehle		
COUNT	(addr1 ===> addr2 +n)	Lesen der Länge, Berechnen des Anfangs
TYPE	(addr +n ===>)	Ausgabe von +n Zeichen ab addr
-TRAILING	(addr +n1 ===> addr +n2)	Abschneiden von Leerzeichen am Ende
EXPECT	(addr +n ===>)	ab addr +n Zeichen einlesen
SPAN	(===> addr)	Anzahl der gelesenen Zeichen in addr
."	(===>)	Ausgabe der Zeichenkette bis "
Konvertierungsbefehle		
<#	(===>)	Bereitstellen einer leeren Kette
#	(+d1 ===> +d2)	Konvertieren einer Stelle
#S	(+d ===> 0 0)	Ausführen von # bis Rest = 0
#>	(32b ===> addr +n)	Abschluß der Konvertierung, Bereitstellen der Ausgabekette
HOLD	(c ===>)	c in Kette einfügen
SIGN	(n ===>)	Vorzeichen von n in Kette einfügen
CONVERT	(+d1 addr1 ===> +d2 addr2)	Konvertieren ab addr1 bis addr2
BASE	(===> addr)	Variable für aktuelle Basis
DECIMAL	(===>)	Zahlenbasis dezimal setzen
HEX	(===>)	Zahlenbasis hexadezimal setzen
weitere Zahlenausgaben		
.R	(n +n ===>)	Anzeige n + n-stellig rechtsbündig
D.R	(d +n ===>)	Anzeige d + n-stellig rechtsbündig
Massenspeicheroperationen		
BLOCK	(u ===> addr)	Lesen von Block u auf addr
BUFFER	(u ===> addr)	Reservieren addr für Block
UPDATE	(===>)	markiert aktiven Puffer als geändert
FLUSH	(===>)	Retten und Leeren aller Puffer
SAVE-BUFFERS	(===>)	Retten aller Puffer
Behandlung des Quelltextstroms		
LIST	(u ===>)	Anzeige Block u
LOAD	(u ===>)	Block interpretieren
-->	(===>)	Weiterladen ab nächstem Block
.(	(===>)	Kommentarausgabe bis)
(<Leerzeichen>	(===>)	Kommentar bis)

Wegbereiter der Informatik



PAFNUTI LWOWITSCH TSCHEBYSCHEW

*** 1821 Okatowo,**
† 1894 St. Petersburg

Der russische Mathematiker P. L. Tschebyschew ist der Gründer der Petersburger Mathematikerschule, aus der zum Beispiel A. A. Markow und A. M. Ljapunow als seine Schüler hervorgegangen sind. Er arbeitete auf verschiedenen Gebieten der Analysis, der Wahrscheinlichkeitstheorie, der Zahlentheorie und der Mechanik.
Tschebyschew stammt aus einer adligen Familie und erhielt seinen Schulunterricht im Elternhaus. 1832 verzog die Familie nach Moskau, um den Söhnen den Besuch der Universität zu erleichtern. Schon 1837 – also mit 16 Jahren – ließ sich Tschebyschew an der physikalisch-mathematischen Fakultät der Moskauer Universität immatrikulieren und beendete sein Studium 1841. Fünf Jahre später erlangte er hier die Magisterwürde und siedelte 1847 nach St. Petersburg über, wo er an der Universität Vorlesungen zur Algebra und zur Zahlentheorie hielt. Im Jahre 1849 verteidigte er seine Dissertation und wurde 1850 Professor an der Petersburger Universität. Dieses Amt bekleidete er bis 1882, danach widmete er sich an der Petersburger Akademie (deren ordentliches Mitglied er 1859 geworden war) ausschließlich seiner wissenschaftlichen Tätigkeit. Nebenher war er Mitarbeiter in verschiedenen Kommissionen bei mehreren russischen Ministerien.
In der mathematischen Wissenschaft, besonders in der Angewandten Mathematik, hat sich Tschebyschew bleibende Verdienste erworben. Bekannt sind die nach ihm benannten Polynome

$T_n(x) = \cos(n \arccos x)$,

die eine besondere Rolle bei der Approximation von Funktionen spielen und einen Sonderfall der Jacobischen Polynome darstellen. Für die numerische Berechnung des bestimmten Integrals einer Funktion fand er die Quadraturformel

$$\int_{-1}^{1} f(x)\,dx \approx \frac{2}{n} \sum_{i=1}^{n} f(x_i),$$

die für Polynome (n – 1)-ten Grades exakt ist (wenn n die Anzahl der Interpolationsknoten bedeutet).
Weniger bekannt geworden ist die Tatsache, daß Tschebyschew auch eine Rechenmaschine konstruiert hat, von ihm Arithmometer (Zahlenmesser) genannt. Sie ist für additive und subtraktive Verarbeitung 10stelliger Zahlen ausgelegt; in ihr ist das Prinzip des gleitenden Zehnerübertrags verwirklicht. Tschebyschew hat das erste Exemplar dieser Maschine dem Pariser Museum für Kunst und Handwerk übergeben und an der dortigen Akademie darüber auch vorgetragen. Ein zweites, verbessertes Exemplar blieb in Petersburg. In der Mechanik hat Tschebyschew eine Theorie der Gelenkmechanismen entwikkelt. Und so entsprang sein Arithmometer wohl hauptsächlich seinem konstruktiven Interesse, den gleitenden Zehnerübertrag zu erproben. Denn die Maschine selbst ist sehr unhandlich und nur mit großem Kraftaufwand zu bedienen, auch fehlt ihr eine Vorrichtung zur Ziffern-Rückstellung oder eine Bedienungskurbel. Es gibt keinerlei Hinweise darauf, daß diese Maschine überhaupt (und sei es von Tschebyschew selbst!) jemals benutzt worden ist.
Tschebyschew, familiär ungebunden, hat zahlreiche Reisen ins Ausland unternommen, sei es, um große Industrieanlagen zu besichtigen oder mit bedeutenden Fachkollegen (Ch. Hermite, J. Bertrand, L. Kronecker) zusammenzutreffen. Er war gewähltes Mitglied der Akademien in Berlin, Bologna, Paris und London.

Dr. Klaus Biener

```
/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\
Ø PROGRAM GRAFIDRU;  (* Nadeldrucker: K631x, EPSON LX-86, usw. *)
Ø (* Compiler-Option: <<O>>     (TURBO Version 3.ØØx - CP/M 2.2)
Ø                        Bei Eingabe von <<C>> noch <<E>> auf den
Ø                        Wert: 7EA9   setzen, damit MicroShell
Ø                        lauffaehig bleibt (C2FF - 1Ø K - 57*128). *)
Ø (* Die X,Y-Werte-Tupel einer ASCII-SDF-Datei mit Definitions-
Ø    /Wertebereich Ø<=X<=1Ø.84 bzw. Ø<=Y<=9.Ø3 werden als Vektor
Ø    (16384 mal 1 Byte) im TPA fuer ein Nadeldrucker-Pixel-Bild
Ø    von einer Breite 512 Sprossen und einer Hoehe 256 Abstands-
Ø    einheiten mittels der nach Pascal umgesetzten  PROCEDURE
Ø    PIXEL aus der Veroeffentlichung: Stamer,E. "Bilderzeugung
Ø    mit Nadeldrucker und PC" in NTB 31(1987)3,S.74-75
Ø    aufgebaut.
Ø Eingabedatei:   (Aufbau analog z.B. in BASIC:  WRITE#1,X,Y )
Ø ============
Ø x,y,[x,y, ...,] ...*<CRLF>
Ø oder:
Ø x,y<CRLF>
Ø ...        ...
Ø *<CRLF>
Ø x,y<CRLF> ...      ... +<CRLF>
Ø Das Ausdrucken des Pixel-Bildes erfolgt mit den als EXTERNAL-
Ø Routinen gestalteten FOTRAN-Subroutinen DRUCK, DRZEIL sowie
Ø SRL, ANDB, die aus obiger Quelle unveraendert uebernommen,
Ø aber auf die externe TPA-Adresse:  7EAAh  gelinkt wurden.
Ø Diese EXTERNAL-Routine DRUCK.OVX wird mit dem RUN-Time-Lader
Ø XPOLADER dynamisch nachgeladen und dann abgearbeitet, wobei
Ø das mittels der Prozedur PIXEL aufgebaute Bild ausgedruckt
Ø wird.
Ø B>A:M8Ø DRUCKA,TTY:=TTY:/Z/X<CR>
Ø             .z8Ø
Ø      title Aufrufvermittler DRUCKA.MAC fuer Subroutine DRUCK
Ø             EXTRN DRUCK
Ø DRUCKA::call DRUCK
Ø             ret
Ø             end<<^Z>>
Ø B>A:F8Ø DRUCK,DRUCK=DRUCK.FOR/N<CR>
Ø B>A:F8Ø DRZEIL,DRZEIL=DRZEIL.FOR/N<CR>
Ø B>A:M8Ø SRL,TTY:=TTY:/Z/X<CR>
Ø SRL::srl (hl)
Ø        ret
Ø        end<<^Z>>
Ø B>A:M8Ø ANDB,TTY:=TTY:/Z/X<CR>
Ø ANDB:ld c,(hl)
Ø        ld a,(de)
Ø        and c
Ø        ret
Ø        PUBLIC ANDB
Ø        end<<^Z>>
Ø B>A:L8Ø /P:7EAA,DRUCKA,DRUCK,DRZEIL,SRL,ANDB,<<^E>>
Ø A:FORLIB/S/E<CR>
Ø B>A:POWER SAVE DRUCK.OVX  7EAA  57<CR>                          *)
Ø LABEL 1Ø;
Ø TYPE Abb=ARRAY[1..16384] OF BYTE;        Str11=STRING[11];
Ø VAR BILD:Abb;          Flag:CHAR;
Ø      f   :Text;        X,Y :REAL;  I,LAD,EAD:INTEGER;
Ø      Name:STRING[14]; DSN:Str11; Ber        :STRING[6Ø];
Ø
Ø PROCEDURE PIXEL (VAR BILD:Abb; VAR X,Y:REAL);
Ø (* TURBO-Pascal-Adaption aus der FORTRAN-Subroutine nach
Ø   Stamer,E.: "Bilderzeugung mit Nadeldrucker und PC"
Ø                NTB 31(1987)3,S.74-75                             *)
Ø CONST BITS:ARRAY[1..8] OF BYTE=($1,$2,$4,$8,$1Ø,$2Ø,$4Ø,$8Ø);
Ø VAR SPROSS,ZZEILE,DZEILE,BITPOS,INDEX:INTEGER;
Ø BEGIN
1 IF (X<Ø.Ø) OR (X>1Ø.84) OR (Y<Ø.Ø) OR (Y>9.Ø3) THEN EXIT;
1 SPROSS:=        TRUNC (X * 47.23)   +1;
1 ZZEILE:= 256 - ROUND (Y * 28.3499) +1;
1 DZEILE:=(ZZEILE -1) DIV 8;
1 BITPOS:= 9 - ( ZZEILE - 8 * DZEILE );
1 INDEX:= DZEILE * 512 + SPROSS;
1 BILD[INDEX]:= BILD[INDEX]  OR  BITS[BITPOS]
1 END;
Ø
Ø PROCEDURE LIESW (VAR X:REAL; VAR Flag:CHAR);
Ø VAR Z:STRING[3Ø];  W:CHAR; K,L:INTEGER;
Ø BEGIN
1   L:=33;  (* Real-String max. 3Ø Zeichen lang *)
1   WHILE (L>Ø) AND (Flag<>'E') DO BEGIN
2   K:=Ø;    Z:='';
2   REPEAT
3    Read(f,W);  K:=K+1;
3    IF W=',' THEN K:=33   ELSE
3    IF (W<>#32)AND(W<>#13)AND(W<>#1Ø)AND(W<>#9)
3                                 THEN   Z:=Z+W
3   UNTIL (K>29) OR (EOLN(f)) OR (EOF(f));
2   IF EOF(f) THEN Flag:='E';
2   VAL (Z,X,L);
2 IF L<>Ø THEN BEGIN Writeln('*** Fehlerhafte Eingabedaten:"',
3 Z,'" ---> Ignore'); Flag:='F' END
2                                     END
1 END;

Ø
Ø PROCEDURE LIESPIXEL (VAR BILD:Abb);
Ø (* Bildaufbau aus der Eingabedatei durch Rueckladen *)
Ø VAR p:FILE; (* Untyped File fuer PIXEL-Bild: 128*128 Byte *)
Ø BEGIN ASSIGN(p,Name); RESET(p);
1 IF FILESIZE(p)<>128 THEN BEGIN
2 Writeln('*** Unmatched Input-File'); HALT END;
1 BLOCKREAD(p,BILD,128); CLOSE(p)
1 END;
Ø
Ø PROCEDURE DRUCK (VAR BILD_Byte:BYTE); EXTERNAL $7EAA;
Ø (* oder: BEGIN INLINE($2A/BILD_Byte/$CD/$AA/$7E)       *)
Ø (* ====         ld hl,(BILD_Byte)                       *)
Ø (*              call  7EAAh                             *)
Ø (*              end                                     *)
Ø (*        END;                                          *)
Ø
Ø (*$I XPOLADER.INC  *)
Ø
Ø BEGIN    (*  ==== Hauptprogramm  ====  *)
1 Writeln(#13#10#10#9#134'PIXEL - Druckprogramm fuer Bildformat'+
1 ' 1Ø,84 cm x 9,Ø3 cm'#132);
1 Write  (             #9'======= 7EA9 ===== V.m=2.1 =========='+
1 '=================='#13#10#1Ø);
1 IF ParamCount=1 THEN Name:=ParamStr(1) ELSE BEGIN
2 Write('ENTER [<d>:]<filename>.<type> der x,y-Werte-Datei ==>');
2 Readln(Name);
2 FOR I:=1 TO LENGTH(Name) DO Name[I]:=UpCase(Name[I])
2                                              END; Flag:='A';
1 (*$I-*)  ASSIGN(f,Name); RESET(f); (*$I+*)
1 IF IORESULT<>Ø THEN BEGIN Writeln;
2 Writeln('*** Eingabedatei ',Name,' nicht gefunden !');HALT END;
1 LIESW(X,Flag); LIESW(Y,Flag); IF (Flag='E') OR (Flag='F') THEN
1 BEGIN
2 Writeln('Eingabedatei leer bzw. unmatched x,y-Tupel');
2 Writeln(
2 'ANNAHME: Eingabedatei ist der DUMP eines BILD-Bereiches'+
2 '  (256/8 * 512 Byte)'+#13#1Ø+'=======  --> Let'+#39+
2 's go to DRUCK.OVX');           LIESPIXEL(BILD);     GOTO 1Ø
2 END;
1 FOR I:=1 TO 16384 DO BILD[I]:=$ØØ;          (* Leeres Bild *)
1 FOR I:=1 TO   512 DO BILD[I]:=$8Ø;      (* Bild-Oberkante  *)
1 FOR I:=15872 TO 16384 DO BILD[I]:=$Ø1;(* Bild-Unterkante *)
1 FOR I:=Ø TO 31 DO  BILD[I*512+1]:=$FF;     (* Linker Rand *)
1 FOR I:=1 TO 32 DO  BILD[I*512]:=$FF;      (* Rechter Rand *)
1 REPEAT               (* Bildaufbau aus der x,y-Eingabedatei *)
2  IF Flag='F' THEN Flag:='A' ELSE
2  PIXEL (BILD,X,Y);
2  LIESW(X,Flag);  LIESW(Y,Flag);
2  IF Flag='F' THEN Flag:='A' ELSE
2  PIXEL (BILD,X,Y);
2  LIESW(X,Flag);  LIESW(Y,Flag)
2 UNTIL Flag='E';
1 1Ø: DSN:='DRUCK   OVX'; LAD:= $7EAA;  EAD:=$FFFF;
1      XPOLADER (DSN,LAD,EAD);(* Druckroutine dynamisch laden *)
1      IF (LAD=Ø) OR (EAD=Ø) THEN BEGIN
2      Writeln;Writeln (DSN,' not found'); HALT END;Writeln(lst);
1      DRUCK (BILD[1]); Writeln(lst); Writeln(lst); Writeln;
1      Writeln('*** '+
1      'Druckergrafik aus der Datei: ',Name,' gedruckt.');Writeln;
1      Write('ENTER Bild-Unterschrift fuer die gedruckte Grafik'+
1      #13#1Ø'==============>');
1      Readln(Ber);Writeln(lst,#9#9,Ber);Writeln(lst);Writeln(lst)
1 END.
```

Bild 3 Grafikprogramm für Nadeldrucker nutzt Fortran- und Assembler-Subroutinen als externe Unterprogramme

```
/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\
Ø PROGRAM GRAFIK;
Ø VAR X,Y:REAL;   I:INTEGER;
Ø      f: Text;
Ø      Name: STRING[14];
Ø BEGIN { main }
1 Name:='GRAFI.DAT';
1 assign(f,Name);  Rewrite(f);
1 X:=Ø;  Y:=Ø;
1 WHILE  X<=1Ø.ØØØ DO BEGIN
2 FOR I:=1 TO 5ØØ DO BEGIN
3 X:=X+Ø.Ø1;  Y:= 9.ØØØ*SIN(X);
3 IF Y<Ø THEN Y:=-Y;
3 Write (f,X:6:3,',',Y:5:3,',')
3                        END { for };
2 Writeln(f,'*');
2 I:=Ø;
2 WHILE I<=1ØØ DO BEGIN
3 I:=I+1; X:=X+Ø.1;
3 IF X>1Ø.ØØØ THEN I:=1Ø1;
3 Y:= 9.ØØØ*COS(X);
3 IF Y<Ø THEN Y:=-Y;
3 Write(f,X,',',Y,',')
3                        END { while };
2 Writeln(f,'+')
2 END { while };
1 CLOSE(f)
[illegible]

* * * * * * * * * * * * *  Ausgegebene Daten:   * * * * * * * * *

 Ø.Ø1Ø,Ø.Ø9Ø, Ø.Ø2Ø,Ø.181, Ø.Ø3Ø,Ø.27Ø, Ø.Ø4Ø,Ø.36Ø, ...
 ...            ...
4.99Ø,8.655, 5.ØØØ,8.63Ø,*
  5.Ø999999990E+ØØ,  3.4Ø17996758E+ØØ,  5.1999999990E+ØØ, ...
  ...                ...
  1.ØØ99999999E+Ø1,  7.Ø251136286E-Ø1,+
\______________________________________________________________________/
```

Bild 4 Programm zur Erzeugung einer ASCII-SDF-Datei und Ausschnitt aus einer Datei GRAFI.DAT

Fortsetzung von Seite 206

(Bild 4) erstellt eine ASCII-SDF-Datei, die von GRAFIDRU (Bild 3) eingelesen und als Pixel-Bild ausgedruckt wird. Mit MicroShell (Copyright © von New Generation Systems) kann die Druckerausgabe auf eine Datei umgeleitet und später auf einem K631x, EPSON LX-86 oder ähnlichen Druckern direkt ausgedruckt werden:

A>SH<CR>
A(0) GRAFIDRU >* B:GRAFI.DMP<CR>
A(0) –x<CR>
A>PIP LST:=B:GRAFI.DMP<CR>

Literatur

/1/ Nerz, H.: Assemblerroutinen in Turbo-Pascal. Francis-Verlag München: mc (1986) 6, S. 78

/2/ Kern, U.: INLASS – der Turbo-Assembler. Chip-Sonderheft Turbo-Pascal, Vogelverlag Würzburg 1986, S. 86

/3/ Hanisch, Ch.: EXEC – ein Startprogramm für dBase II/REDABAS. MP, Berlin 2 (1988) 11, S. 131

/4/ Hanisch, Ch.: Automatische Generierung von BASIC-INLINE-POKEs. MP 3 (1989) 3, S. 75

/5/ Stamer, E.: Bilderzeugung mit Nadeldrucker und PC. NTB 31 (1987) 3, S. 74

Entwicklung zuverlässiger Software in der Gerätetechnik

Uwe Schneider
Technische Hochschule Ilmenau, Sektion Gerätetechnik

Einführung

Das Einsatzspektrum von Mikrorechnern in der Gerätetechnik erweitert sich progressiv. Mit ihrem Einsatz können bekannte Gerätefunktionen effektiver bzw. erstmals zweckmäßig realisiert werden. Noch bedeutungsvoller ist die Möglichkeit der Realisierung völlig neuartiger Funktionen und Konstruktionsprinzipien. Beispiele sind automatisierte und optimierte Abläufe von Gerätefunktionen, Selbstdiagnose, automatische Fehlererkennung und -beseitigung sowie integrierte Antriebe.

Mikrorechner werden praktisch in allen Geräteklassen eingesetzt (Bild 1).

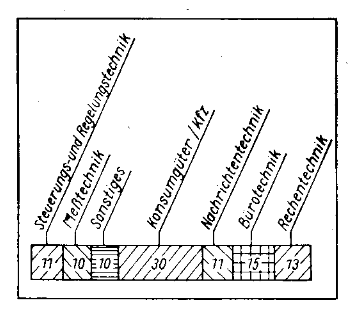


Bild 1 Einsatzgebiete von Mikrorechnern /1/

Im Beitrag wird vorzugsweise auf die Einsatzgebiete Steuerungs-, Regelungs- und Meßtechnik als Bestandteile der Antriebstechnik eingegangen.

Eine Untersuchung verschiedener Steuerungssysteme zeigt /2/, daß die Modularisierung der Hardware weit fortgeschritten ist, während kaum anwenderspezifische Software angeboten wird (Tafel 1). Durch die zunehmende softwaremäßige Realisierung von Gerätefunktionen wächst der Softwarebedarf, so daß effektive Programmiermethoden ein unbedingtes Erfordernis sind.

Tafel 1 Softwareangebot zu modularen Steuersystemen /2/

Bezeichnung	Anbieter	Software
Modulares Steuersystem	TH Ilmenau Technikum Suhl	MIBS minimale Betriebssystemfunktionen (2 KByte) CP/M
P6000	VEB Robotron-Rationalisierung Weimar	keine
Modulares Steuersystem	VEB Kombinat Carl Zeiss JENA (kombinatsintern)	UDOS-ähnliches Betriebssystem (12 KByte)
NANOS	IHS Warnemünde/Wustrow	CP/M DIALOG 880
SMS-Baugruppensortiment	AdW der DDR	keine

Um die damit zusammenhängenden Probleme lösen zu können, ist es zunächst einmal notwendig, die Eigenschaften gerätetechnischer Software zu analysieren.

Eigenschaften gerätetechnischer Software

Gerätechnische Software zeichnet sich in der Regel durch Echtzeitfähigkeit aus. Vom Charakter des programmierten Problems hängt ab, welche Zeitforderungen sich aus dem Echtzeitbetrieb ergeben. Diese Zeitforderungen sind in der Antriebstechnik besonders hoch. So sind Abtastzeiten von 5 ms bei hochdynamischen Antrieben nicht ungewöhnlich. Die Echtzeitfähigkeit ist auch eine Voraussetzung für die Zuverlässigkeit, die ja maßgeblich von der Software bestimmt wird.

Zuverlässigkeit gerätetechnischer Software bedeutet neben der als selbstverständlich vorauszusetzenden Ablauf- und Benutzungssicherheit, daß zu jedem Zeitpunkt die Kontrolle über das zu steuernde System gewährleistet sein muß. Die Zuverlässigkeit erweist sich demzufolge erst im Zusammenspiel von Software und gerätetechnischer Peripherie. Sie ist Voraussetzung für Arbeitsschutz und Gerätesicherheit, da unvorhergesehene Zustände und Programmfehler unmittelbar materiellen Schaden anrichten können.

Zuverlässigkeit ist eine unabdingbare Forderung. Sie wird erreicht durch die Anwendung von Methoden der Fehlervermeidung im Entwurf, der Fehlerbeseitigung in der Testphase sowie der Fehlererkennung und -umgehung im Betrieb. Diese Aussage verdeutlicht bereits die Bedeutung, die einer Entwurfsmethodik und Entwicklungsumgebung zukommt, die Fehler vermeidet.

Für gerätetechnische Software ist ein umfangreicher Informationsaustausch zwischen Mikrorechner und gerätetechnischer Peripherie charakteristisch. Insbesondere sind die Rückmeldungen von der Peripherie an den Mikrorechner aus zwei Gründen von Bedeutung:

Aufgrund der unterschiedlichen Arbeitsgeschwindigkeiten von mechanischem System und digitalem Rechner ist eine Synchronisation nötig.

Des weiteren – und das ist wiederum für die Zuverlässigkeit von Bedeutung – muß der Rechner Rückmeldungen über die exakte Ausführung angewiesener mechanischer Aktionen erhalten. Der Austausch von Meldungen, die aktiv den Programmablauf beeinflussen, zwischen Mikrorechner und gerätetechnischer Peripherie ist nur effektiv durch ein leistungsfähiges Interruptsystem zu realisieren.

Gerätetechnische Software zeichnet sich durch starke Applikationsbezogenheit aus. Sie ist im Gegensatz zu Anwenderprogrammen für wissenschaftlich-technische Aufgaben, Textverarbeitungssoftware usw. nicht universell nutzbar, sondern an Funktion und Konfiguration des konkreten Gerätes gekoppelt. Ihre Verbreitung hängt von den Stückzahlen des Gerätes ab. Das hat Konsequenzen für den Aufwand bei der Entwicklung der Software.

Charakteristische Eigenschaften gerätetechnischer Software sind:

- Echtzeitbetrieb
- Interruptverarbeitung
- hohes Maß an Informationsverarbeitung
- starke Applikationsbezogenheit.

Die Problemsituation der Softwareproduktion

Als Folge der dargestellten Eigenschaften gerätetechnischer Software entstehen zumeist sehr spezialisierte Softwarelösungen auf Assemblerniveau, die sich jeder Anwender für oft gleichartige Probleme immer wieder neu schafft. Diese Situation ist in mehrfacher Hinsicht unbefriedigend.

Es ist volkswirtschaftlich uneffektiv, wenn für gleichartige Problemstellungen ständig neue Lösungen gesucht werden. Das ist insbesondere der Fall, wenn man berücksichtigt, daß der große Bedarf an Software von einer zu geringen Zahl qualifizierter Programmierer bewältigt werden muß. Der Programmierer gerätetechnischer Software muß sowohl über Kenntnisse auf dem Gebiet der Gerätetechnik als auch der Informatik verfügen. Da das sicher nicht immer in ausreichendem Maße der Fall ist, wird mit der wiederholten Lösung eines Problems nicht zwangsläufig ein Qualitätszuwachs verbunden sein.

Die Situation ist weiterhin unbefriedigend, weil der vorzugsweise Einsatz der Assemblerprogrammierung, gerade im Hinblick auf steigernde Komplexität der Programme und eine größere Prozessorvielfalt, der Mehrfachnutzung existierender Problemlösungen im Wege steht.

Assemblerprogramme sind oft sehr spezialisiert, wenig strukturiert und schlecht dokumentiert, so daß es meistens einfacher ist, ein neues Programm zu schreiben als ein vorhandenes zu ändern. Assemblerprogramme sind nicht portabel. Die Effektivität der Programmentwicklung ist gering. Mit steigender Komplexität der Programme wird die Sicherung der Zuverlässigkeit problematisch. Viele Fehler werden erst in der Testphase entdeckt. Die effektive Programmierung zuverlässiger Software, die an gleichartige Aufgaben leicht angepaßt werden kann, wird durch die ausschließliche Verwendung der Assemblerprogrammierung behindert.

Hinzu kommt ein Problem, das daraus erwächst, daß gerätetechnische Software immer an eine konkrete gerätetechnische Peripherie gebunden ist. Eine zentrale Programmbibliothek, in der jeder Anwender die Software für sein Problem findet, ist aus diesem Grund nicht möglich.

Aufgabenspezifische Entwicklungsumgebungen als Möglichkeit der effektiven Softwareproduktion

Zum Inhalt des Begriffs Entwicklungsumgebung

Eine mögliche Lösung des Problems besteht in der Schaffung geeigneter Entwicklungsumgebungen, die die programmtechnische Umsetzung der zu realisierenden Funktionen bestimmter Geräteklassen unterstützen.

Eine höhere Programmiersprache, erweitert um eine problemorientierte Sprachebene und in Verbindung mit die Programmentwicklung und den Programmtest unterstützenden Softwarewerkzeugen stellt eine Entwicklungsumgebung dar. Eine solche Entwicklungsumgebung bietet die Möglichkeit, mit geringem eigenem Entwicklungsaufwand in Verbindung mit entsprechenden Hardware-

modulen komplette Anwenderlösungen aufbauen zu können.
Die benutzer- bzw. problemorientierte Sprachebene wird durch eine Menge von Softwaremodulen repräsentiert. Da sie eine Erweiterung der Sprache sind, können die Module auch als Elemente einer Fachsprache aufgefaßt werden. Die Entwicklungsumgebung vereint in sich jedoch mehr als nur eine Fachsprache.
Der Unterschied besteht zum einen in dem nach oben offenen Konzept der Entwicklungsumgebung. Das heißt, der Anwender kann nicht nur über die Elemente beider Sprachebenen verfügen. Er hat vielmehr die Möglichkeit, mit den Mitteln der Sprache selbst die Ebenen um neue Elemente zu erweitern bzw. neue Sprachebenen zu entwikkeln. Zum anderen bietet eine Entwicklungsumgebung im Gegensatz zur Fachsprache vielfältige Unterstützung in Form von Softwarewerkzeugen.

Programm- und gerätetechnische Forderungen an die Softwaregestaltung

Die die benutzerspezifische Sprachebene bildenden Softwaremodule müssen bestimmten programm- und gerätetechnischen Anforderungen genügen.
Softwaremodule sind Bausteine mit definierten Schnittstellen, die unter bestimmten Bedingungen festgelegte Funktionen erfüllen. Schnittstellen und Funktion müssen eindeutig sein, damit die Softwaremodule zu einem Programmsystem verbunden werden können. Von Bedeutung ist die funktionelle Abgrenzung der Module, bei der das Systemkonzept berücksichtigt werden muß. Das heißt, die Module müssen funktionell aufeinander abgestimmt sein.
Die programmtechnischen Forderungen umfassen die Merkmale „guter Software". Das sind im einzelnen Strukturierung, Leistungsfähigkeit und Portabilität. Zu diesen Kriterien guter Programme gibt es eine umfangreiche Literatur /4/, /5/, so daß hier nur auf zwei Aspekte näher eingegangen wird.
Strukturierte Programme sind durch eine bestimmte innere Struktur und die funktionelle Abgrenzung gegen andere Programmabschnitte gekennzeichnet. Durch Strukturierung entstehen übersichtliche, durchschaubare Programme. Alle Phasen des Softwarelebenszyklus vom Entwurf bis zur Wartung können so effektiver gestaltet werden. Die verfügbaren Programmiersprachen unterstützen in unterschiedlich starkem Maße die strukturierte Programmierung.
Als zweiter sehr wesentlicher Punkt sei die Portabilität genannt. Portable Software läuft auf möglichst vielen unterschiedlichen Rechnern. Dazu muß sie maschinenunabhängig und streng genommen autark sein. Die Portabilität ist von entscheidender Bedeutung für Austauschbarkeit und Verbreitung. Nur portable Software überdauert Rechnergenerationen.
Die gerätetechnischen Anforderungen an die Softwaremodule ergeben sich aus der Verbindung von Digitalrechner und Gerätetechnik. Bei diesen Forderungen handelt es sich um die eingangs erwähnten Punkte Echtzeitfähigkeit und Informationsaustausch zwischen Rechner und gerätetechnischer Peripherie. Der Kern der gerätetechnischen Forderungen besteht darin, daß der Rechner zu jedem Zeitpunkt die Kontrolle über die gerätetechnische Peripherie besitzt. Dazu sind neben einer ausreichenden Rechenkapazität genügend Informationen über den Zustand des gerätetechnischen Systems erforderlich.

Wahl der Programmiersprache

Ein Problem stellt die Wahl einer geeigneten Programmiersprache zur Realisierung der Softwaremodule dar. Aus dem bisher gesagten wurde deutlich, daß der Einsatz höherer Programmiersprachen unumgänglich ist, wenn zuverlässige Software effektiver programmiert und einem breiten Nutzerkreis erschlossen werden soll. Aus den Anforderungen an gerätetechnische Software lassen sich die Forderungen bezüglich der Sprache ableiten. Sie sollte die strukturierte Programmierung mindestens unterstützen oder besser erzwingen. Sprache und Programme müssen in hohem Maße portabel sein. Die Sprache muß die Merkmale Echtzeitfähigkeit und nach Möglichkeit Multitaskfähigkeit besitzen. Die Interruptverarbeitung muß uneingeschränkt realisierbar sein. Der Programmierer muß die Möglichkeit der problemorientierten und der maschinenorientierten Arbeit besitzen. Schließlich sollte es auf unkomplizierte Weise möglich sein, vorhandene Assemblerprogramme einzubinden. Letztendlich spielen Kriterien eine Rolle, die vom Grad der Eignung einer Sprache unabhängig sind. Wichtigster Punkt ist die Bewährtheit und Verfügbarkeit einer Sprache durch Lehrmaterialien, Softwarewerkzeuge, Nutzergemeinschaften usw.
Subjektive Momente spielen bei der Wahl einer Programmiersprache ebenfalls eine nicht zu unterschätzende Rolle.
Höhere Sprachen, die den Anforderungen der Programmierung von Anwendersoftware in der Gerätetechnik genügen, sind die bisher vorwiegend in der Systemprogrammierung eingesetzte Sprache C und die speziell für steuerungstechnische Zwecke geschaffene Sprache Forth.
Ohne auf die Sprache näher eingehen zu wollen, werden an dieser Stelle einige Anmerkungen zu Forth gemacht. Es handelt sich bei Forth um keine Sprache im herkömmlichen Sinne: Forth vereinigt in sich das Sprachkonzept, einen Compiler, einen Editor, einen Interpreter, Betriebssystemfunktionen sowie optional einen Editor und Assembler. Alle diese Komponenten sind weitestgehend in Forth programmiert.
Einzigartig ist die Möglichkeit der Erweiterung der Sprache um neue Elemente, die selbst in Forth programmiert werden können. Damit ist Forth hervorragend zur Programmierung problemorientierter Sprachebenen, das heißt Fachsprachen, geeignet. Oder anders formuliert: Mit Forth lassen sich aufgabenspezifische Entwicklungsumgebungen, wie sie als Möglichkeit der Lösung des Problems der effektiven Softwareproduktion in der Gerätetechnik skizziert wurden, günstig realisieren. Zudem stellt Forth an sich bereits eine universelle Entwicklungsumgebung dar, die relativ leicht auf den verschiedensten Rechnern implementiert werden kann und die für den Programmierer sehr transparent ist (Bild 2).
Als Beispiel einer solchen Entwicklungsumgebung für den Aufgabenbereich Bilderkennung sei das Bilderkennungssystem BES 2000 des VEB Studiotechnik Berlin genannt /6/. In /7/ wird ein Forth-Programm zur Steuerung mehrerer 4-Phasen-Schrittmotoren vorgestellt.
Forth hat in der DDR eine größere Verbreitung, als das Zeitschriftenstudium vermuten

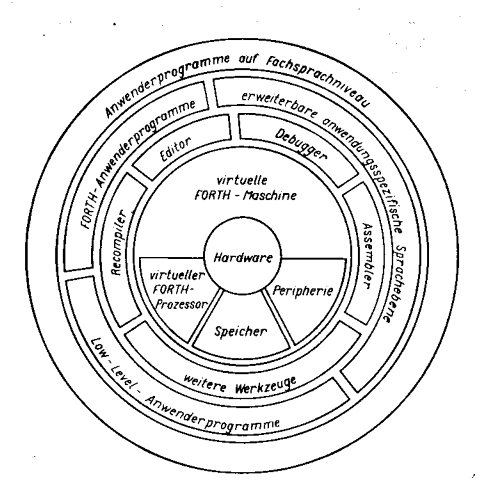


Bild 2 Entwicklungsumgebung für gerätetechnische Antriebe

läßt. Forth-Versionen existieren für 8- und 16-Bit-Systeme. International ist bereits ein Forth-Prozessor verfügbar, der Forth-Code direkt ausführen kann /3/. Forth ist also durchaus eine lebendige Sprache, eine Sprache mit Zukunft.
Diese Ausführungen sind als Anregungen zu betrachten, bei der Gestaltung neuer Software über die unmittelbaren Tagesbedürfnisse hinaus zu denken und hierbei mehr konzeptionell unter dem Gesichtspunkt einer hohen volkswirtschaftlichen Effektivität zu arbeiten.

Literatur

/1/ Werner, D.: Programmierung von Mikrorechnern. VEB Verlag Technik, Berlin 1986
/2/ Michael, K.: Steuersystem für modulare Antriebe. Diplomarbeit, Technische Hochschule Ilmenau, Sektion Gerätetechnik, 1987
/3/ Forth-Worte werden Fleisch. elektronikpraxis (1986) 4, S. 76
/4/ Nettesheim, H.: Programmentwurf und Programmdokumentation. Düsseldorf 1982
/5/ Kopek, H.: Software-Zuverlässigkeit. Leipzig 1977
/6/ Anwendungsbeschreibung Systemsoftware für Bilderkennungseinheit BEE 1010. RFT-VEB Studiotechnik Berlin.
/7/ Berg, K. P.; Kaiser, R. E.: Stepper Motor Actuation by FORTH. Instruments and Computers, Juli/August 1985, S. 11

✉ KONTAKT ☏

Technische Hochschule Ilmenau, Sektion Gerätetechnik, WB Informationsgerätetechnik, PSF 327, Ilmenau, 6300; Tel. 74539

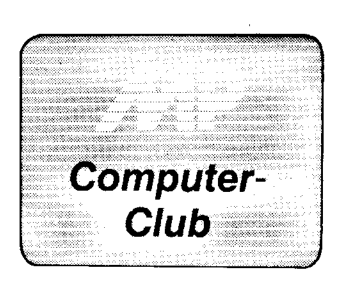
TERMINE

12. Internationales Seminar über Datenbankbetriebssysteme

WER? Tsentrprogrammsystem, Kalinin, UdSSR
WANN? 17. bis 20. Oktober 1989
WO? Suzdal, UdSSR
WAS?
- Datenbanktheorie, Modelle und Werkzeuge
- Methodologien und Hilfsmittel zum Datenbankentwurf
- verteilte Datenbanken
- Datenbankmaschinen
- Wissensbasis-Systeme
- Datenbanken und Expertensysteme
- Datenbanken in CAD/CAM-Systemen
- anspruchsvolle Anwendungen

WIE?
- Veranstaltungssprache: Englisch
- nähere Informationen über: VEB LfA Berlin, Jacques-Duclos-Straße 50/52, Berlin, 1156; Tel. 37 80 35 18

Dr. Lange/Dr. Oßwald



Abfrage des Druckerstatus am AC A 7100

Wer hat es noch nicht erlebt, ein sorgfältig gestaltetes Menübild am AC A 7100 wird durch die Systemmeldung auf dem Bildschirm "list device not ready type R to retry or..." verunstaltet. Und drückt man dann noch versehentlich eine falsche Taste, kann das zu undefinierten Programmabbrüchen führen. Der Grund dafür ist ein ausgeschalteter bzw. nicht betriebsbereiter Drucker.
Dieses Problem ist lösbar, indem man vor dem ersten Druckbefehl den aktuellen Druckerstatus festellt. Dazu werden in den vorgeschlagenen Programmvarianten zwei Komponenten benötigt. Als erstes muß die Portadresse der Schnittstelle zum Drucker bekannt sein. Da beim AC A 7100 das Centronics-Interface in erster Linie als Schnittstelle für den Drucker benutzt wird, ist die Statusabfrage genau auf dieses Interface bezogen und wird über das Port A (Adresse 00C8H) des PPI /1/ vorgenommen. Eine Anpassung an eine andere Schnittstelle ist kein Problem und kann den jeweiligen Systemunterlagen entnommen werden. Die zweite Komponente besteht in einer geeigneten Möglichkeit, das bekannte Port zu lesen.
Mit solchen komfortablen Sprachen wie Basic (Bild 1) oder Turbo-Pascal (Bild 2) ist dies kein Problem, da Standardprozeduren zum Lesen eines Ports bereits implementiert sind. In REDABAS ist dies problematischer. Für diesen Fall wurde ein kleines Assemblerprogramm geschrieben, welches das Lesen des Ports übernimmt (Bild 3).
Es wird in einen Speicherbereich geladen, welcher von REDABAS nur selten erreicht wird. Nur nach einem ausgeführten SORT-Befehl sollte man die Assemblerroutine neu POKEn. Um nicht den ohnehin geringen Speicherplatz für Variablen (und deren Anzahl zu belasten, wird auf die Übergabe in einer Variablen verzichtet (dies ist natürlich auch in den beiden anderen Programmen möglich).
Alle 3 Testprogramme sind so aufgebaut, daß bei betriebsbereitem Drukker das Hauptprogramm ohne Unterbrechung weiterläuft und eine Druckerausgabe erfolgen kann. Sollte aber vor oder während der Abarbeitung des Testprogrammes ein Zustand am abgefragten Ports eintreten, der nicht dem Wert für Betriebsbereitschaft des Druckers entspricht, wird das Programm unterbrochen und mit einer Bildschirmausschrift (die entsprechend positioniert sein sollte) der Fehler mitgeteilt. Das Programm kann erst fortgesetzt werden, wenn die Betriebsbereitschaft des Druckers hergestellt ist. Da der Fehlertext ständig neu geschrieben wird, sollte der Cursor ausgeschalten werden.
Die Programme sind natürlich noch ausbaufähig. Man kann dem Bediener auch die Möglichkeit eines Programmabbruchs geben oder die verschiedenen Zustände des Druckers auswerten. In Tafel 1 sind als Anregung dazu die Ergebnisse der mir zur Verfügung stehenden Druckertypen dargestellt (Basic-Funktion WERT= INP(Portadresse)). Es sei noch darauf hingewiesen, daß in der Zeit vom Netzzuschalten bis zur Betriebsbereitschaft des Druckers kurzzeitig mehrere Zustände am Port des Interfaces anliegen. Die Programme laufen unter SCP 1700 und CP/K 86.

Dirk Pietzsch

```
10    REM ** Hauptprogramm **
20    DEF FNCP$(Z%,S%)=CHR$(27)+"["+FNH$(Z%)+";"+FNH$(S%)+"H"
30    DEF FNH$(N%)=MID$(STR$(N%),2)
40    PRINT CHR$(12) : REM * BS-Loeschen *
100   REM ** Druckerausgabe **
110   GOSUB 10000
1000  END
10000 REM ** UP-Druckerstatus **
10005 PRINT CHR$(27)+"[?141" : REM * Cursor aus *
10010 DROK=INP(200)
10020 IF DROK=198 THEN 10050
10030 PRINT FNCP$(23,15);"Drucker nicht betriebsbereit !"
10040 GOTO 10010
10050 PRINT FNCP$(23,15);"                              "
10055 PRINT CHR$(27)+"[?14h" : REM * Cursor ein *
10060 RETURN
```

Bild 1 Ermitteln Druckerstatus in Basic – AC A 7100

```
PROGRAM Hauptprogramm;

PROCEDURE DRST ; (* UP DRuckerSTatus *);

VAR DR_OK : INTEGER;

BEGIN
  WRITE(^['[?141'); (* Cursor aus *);
  DR_OK:=(PORT[200]);
  WHILE DR_OK <> 198 DO BEGIN
    GOTOXY(15,23);WRITE('Drucker nicht betriebsbereit !');
    DR_OK:=(PORT[200])
  END;
  GOTOXY(15,23);WRITE('                              ');
  WRITE(^['[?14h'); (* Cursor ein *)
END;

BEGIN
  WRITE(#12); (* BS-Loeschen *);
  DRST;       (* Begin Druckerausgabe *)

END.
```

Bild 2 Ermitteln Druckerstatus in Turbo-Pascal – AC A 7100

```
* Hauptprogramm
* Laden Assemblerprogramm
SET CALL TO 60000
POKE 60000,187,104,234,228,200,136,  7,195

** UP-DRuckerSTatus **
@ 0,0 SAY CHR(27)+"[?141"
CALL
DO WHILE PEEK(60008) <> 198
  @ 23,15 SAY 'Drucker nicht betriebsbereit !'
  CALL
ENDDO
@ 23,15 SAY
@ 0,0 SAY CHR(27)+"[?14h"
** ENDE DRST **

RETURN
```

Bild 3 Ermitteln Druckerstatus in REDABAS – AC A 7100

Tafel 1 Ergebnisse der Portabfrage in einigen Zuständen von zwei Druckertypen

Druckerstatus	K 6313/14	LX 800
Drucker ein/betriebsbereit	198	198
Drucker ein/nicht betriebsbereit	197/193	195
Drucker ein/Papierende	201	203
Drucker aus/kein Drucker vorhanden	207	207

Literatur

/1/ Autorenkollektiv: Betriebsdokumentation AC A 7100 – Rechner und Geräte, Dresden: VEB Robotron-Elektronik Dresden

---

Bildschirminhalt auf Kassette speichern

Unter der Rubrik Computerclub wurde in der MP 9/87 ein Vorschlag zum Abspeichern des Bildschirminhaltes des KC 85/3 auf Kassette gemacht.
Mit dem KC 85/1 erreicht man dasselbe bei Nutzung des hier vorgestellten BASIC-Programms.

Vorgehensweise:
Programm eingeben. (Die Ziffern in den Zeilen 2 bis 28 sollen die genaue Länge der Zeilen gewährleisten. Das geht auch anders, bis einschließlich Zeile 30 aber muß jedes Byte sitzen!) Sicherheitshalber abspeichern.
Mit RUN laufen lassen. Dabei werden zunächst nur die MC-Programme in die Zeilen 2 bis 4 gepoket.

Ausprobieren der Teilprogramme:
MC-1 mit CALL 1056 schnelles CLS
MC-2 mit CALL 1088 Bildinhalt gelangt in Zeilen 5, 6, 7 ... 28
MC-3 mit CALL 1130 Umkehrung zu MC-2.

Abspeichern des in die 24 Basic-Zeilen transportierten Bildes erfolgt mit:
POKE 2294, 0: POKE 2295,0:LIST# 1"Name"5

Geladen wird ein abgelegtes Bild mit:
LOAD#1 "Name"

```
PROGRAMM

1 POKE2294,252:POKE2295,8:GOTO30
2 !12345678901234567890123456
3 !12345678901234567890123456789012345 6
4 !1234567890123456789012345678901234 5
5 PRINT"123456789012345678901234567890 12
34567890
            .
            .
            .
28 PRINT"1-----------------------------
---------0
30 !
1000 FORX=1056TO1165:READY:IFY=999THENST
OP
1010 POKEX,Y:NEXTX
1020 DATA229,197,213,33,1,236,43,54,32,6
2,4,6,240,35,54,32,16,251,61,32
1030 DATA246,209,193,225,201,32,0,101,4,
3,0,156,229,197,213,33,148,4
1040 DATA17,2,236,27,27,14,24,6,40,26,11
9,19,35,16,250,35,35,35,35,35,35
1050 DATA35,13,32,238,209,193,225,201,32
,0,142,4,4,0,156,229,197,213,17
1060 DATA148,4,33,2,236,43,43,14,24,6,40
,26,119,19,35,16,250,19,19,19,19
1070 DATA19,19,19,13,32,238,209,193,225,
201,999
```

Bild 1 Programm zum Abspeichern des Bildschirminhalts

```
      E5     PUSH HL        Reg.retten
      C5     PUSH BC
      D5     PUSH DE
      2101EC LD   HL,EC01 Bildanfg.in HL
      2B     DEC  HL        nie 0 in Zeile
      3620   LD   HL,(20) Leerz.
      3E04   LD   A,04      4*240=960
LP1   06F0   LD   B,F0
LP2   23     INC  HL        Naechste Stelle
      3620   LD   (HL),20 Loeschen
      10FB   DJNZ LP2       Block wdh.
      3D     DEC  A         Naechster Block
      20F6   JR   NZ,LP1
      D1     POP  DE        Reg.herstellen
      C1     POP  BC
      E1     POP  HL
      C9     RET
      20                    Leerzeichen
      00                    Zeilenende Z.2
      6504                  Pointer    Z.3
      0300                  Z.-Nr.     Z.3
      9C                    REM-Token  (!)
```

Bild 2 Schnelles Bildschirmlöschen

```
      E5     PUSH HL
      C5     PUSH BC
      D5     PUSH DE
      219404 LD   HL,0494 BASIC-Pu.Anfang
      1102EC LD   DE,EC02 Bildsp.  Anfang
      1B1B   DEC  DE ..     Nie 0 in Zeile
      0E18   LD   C,18      24 Zeilen
LP1   0628   LD   B,28      Je 40 Zeichen
LP2   1A     LD   A,(DE)    Z. aus Bild
      77     LD   (HL),A    In BASIC-Zeile
      13     INC  DE        Naechtes Z.
      23     INC  HL
      10FA   DJNZ LP2
      232323 INC  HL ..     Ueberspringen
      232323 INC  HL ..     Z.-End,Pointer,
      23     INC  HL        Z.-Nr.u.PRINT"
      0D     DEC  C         Naechste Zeile
      20EE   JR   NZ,LP1
      D1C1E1 POP  ......    Reg. herstellen
      C9     RET
      20008E                Wie MC-1
      0404009C
```

Bild 3 Bildinhalt in die Zeile 5 bis 28 übernehmen

```
      E5C5D5 PUSH ......    Reg. retten
      119404 LD   DE,0494
      2102EC LD   HL,EC02
      2B2B   DEC  HL ..
      0E18   LD   C,18
LP1   0628   LD   B,28
LP2   1A     LD   A,(DE)
      77     LD   (HL),A
      13     INC  DE
      23     INC  HL
      10FA   DJNZ LP2
      131313 INC  DE ....
      131313 INC  DE ....
      13     INC  DE
      0D     DEC  C
      20EE   JR   NZ,LP1
      D1C1E1 POP  ......
      C9     RET
```

Bild 4 Umkehrung zu MC-2

Jetzt die Zeilen 1000 bis 1070 löschen. Sie werden nicht mehr benötigt. Danach das fertige Programm abspeichern und die Kopie in Zukunft verwenden.
Da die MC-Programme in den Basic-Zeilen 2 bis 4 abgelegt wurden, stehen sie nach jedem Laden sofort zur Verfügung, was sehr vorteilhaft ist. Allerdings geht jetzt das Listen etwas ungewöhnlich vor sich. Aber wer listet sich schon ständig durch die Zeilen 1 bis 28? Besser man verwendet gleich LIST 30.

Erklärungen:
Einige Computer verfügen über Basic-Befehle, die ein sofortiges Abspeichern beliebiger Speicherbereiche (also auch des Bildspeichers) auf Kassette gestatten. Die KC-Rechner können das aus dem Basic heraus nicht. Es gibt aber die Möglichkeit, Teile des Basic-Programms auszugeben.
Der Bildinhalt wird deshalb in Teile zerlegt und mit MC-2 in vorbereitete Basic-Zeilen abgelegt.
Da der Bildschirm aus 24 Zeilen zu je 40 Zeichen aufgebaut ist, werden entsprechende Basic-Zeilen (5, 6, 7 ... 28) benötigt.
Das Prinzip geht auf das Abspeichern von MC-Programmen in REM-Zeilen zurück. REM ist aber für unser Anliegen nicht brauchbar, da dann beim LIST die Grafikzeichen als Basic-Token interpretiert werden. Deshalb also PRINT" in den Zeilen 5 bis 28. Das hintere Anführungszeichen weggelassen.
Nach CALL 1088 ist der Basic-Bildschirm-Puffer gefüllt und kann mit dem LIST-Befehl auf Kassette gebracht werden. Beim Laden des Bildes mit LOAD listet der Interpreter zwangsläufig und geht anschließend in den Kommandomodus. Jetzt kann mit CALL 1130 der Bildschirmspeicher mit dem eben geladenen Inhalt der Basic-Zeilen gefüllt werden.
Das Programm ist eigentlich als Vorsatz gedacht. Ab Zeile 31 kann deshalb unter Beachtung des unter Vorgehensweise gesagten weiterprogrammiert werden (z. B. Aufbau des Bildes). Eine Taste wird geopfert, damit nach ihrer Betätigung MC-2 aufgerufen wird. Dann hat man den Bildinhalt ohne störende Kommandos auf Kassette.
Wozu ist Zeile 1 gut?
Hat man ab Zeile 31 weiterprogrammmiert, würde der Befehl LIST#1"Name"5 alle Zeilen ab 5 bis zum Basic-Ende auf Kassette auslagern und nicht nur die bis 28, sehr ärgerlich ...
Deshalb die beiden POKEs vor diesem Befehl. Diese setzen den Zeilenpointer der Zeile 30 auf 0. Mit dem Zeilenendezeichen der Zeile 28 stehen dann drei Nullen hintereinander, was Basic-Ende bedeutet.
Da der Interpreter nach LIST anhält, muß mit RUN neu gestartet werden. Jetzt wäre das scheinbare Basic-Ende zur Falle geworden, wenn nicht Zeile 1 die richtigen Werte für den Pointer wieder einstellen würde.

Henrik Luther

Strings plotten in Großschrift

Mit diesem Programm wird dem Basicprogrammierer die Möglichkeit gegeben, Zeichenketten in beliebiger Größe an jeder beliebigen Position des Bildschirms auszugeben. Im Gegensatz zum PRINT besteht keine Bindung an das 8 × 8 Pixelraster.
Zur Benutzung sind folgende Schritte im Basicprogramm notwendig:

1. Ablegen des auszugebenden Strings in der Variablen A$ (z. B. A$ = "Strings plotten")
2. auf Speicherplatz 5 und 6 die Vergrößerungsfaktoren (Höhe, Länge) ablegen (z. B.
POKE 5,1 :REM normale Höhe)
POKE 6,2 :REM doppelt so lang)
3. linke obere Ecke des ersten Zeichens von A$ (10,200) und Farbe(7) festlegen (z. B. PRESET 10,200,7)
4. Programm aufrufen (z. B. CALL <Startadresse>)

Aufgrund seiner Länge (283 Byte) ist dieses Programm nicht zum Einlesen von Basic aus geeignet, da es auf diese Art zweimal Speicherplatz belegen würde.
Es empfiehlt sich eine Eingabe über Modify mit einem nachfolgenden Sichern auf Magnetbandkassette.
Dieses Programm konnte auf Grund seiner großen Sprungdistanzen nicht in verschiebbarem Maschinencode geschrieben werden, so daß bei Verschiebungen eine Anpassung der Sprungadressen erfolgen muß. Im konkreten Fall geschieht das nach Eingabe des Programms durch folgende Basic-Befehle:

DOKE<Startadresse dez. + 127>, <Startadresse + 212>
DOKE<Startadresse dez. + 198>, <Startadresse + 60>

In einem kurzen Beispielprogramm (Bild 1), wird die Anwendung dieser Routine verdeutlicht. Voraussetzung für eine einwandfreie Funktion ist ein korrekt abgelegter Maschinencode ab Adresse 08H (siehe Bild 2).

Thorsten Noske

```
10  WINDOW 0,31,0,39:CLS
20  A$="U 880"
30  FOR K=1 TO 10
40  FOR I=1 TO 10
50  POKE 5,K: POKE 6,I
60  PRESET 10,200,7
70  CALL 8
80  PAUSE 10
90  CLS
100 NEXT I,K
```

Bild 1 Beispiel für Großschrift

0008	CD	18	F0	C5	D5	E5	F5	FD
0010	E5	2A	D7	03	3E	80	ED	A1
0018	20	FC	7C	FE	3A	D0	00	00
0020	3E	41	BE	28	04	00	00	18
0028	EB	23	7E	32	00	00	23	7E
0030	32	01	00	23	7E	32	02	00
0038	23	7E	32	03	00	3A	00	00
0040	47	2A	02	00	7E	E5	FE	60
0048	38	02	D6	20	D6	1F	C5	47
0050	3E	00	C6	08	FE	F8	28	07
0058	10	F8	26	00	6F	18	0E	21
0060	E8	00	A4	04	C6	08	10	FC
0068	11	00	00	5F	19	ED	5B	A6
0070	B7	19	11	08	00	ED	52	0E
0078	08	ED	5B	D3	B7	ED	53	07
0080	00	E5	7E	06	08	17	CD	DC
0088	00	10	FA	2A	07	00	22	D3
0090	B7	3A	05	00	47	3A	D5	B7
0098	00	90	32	D5	B7	E1	23	0D
00A0	79	FE	00	20	DC	3A	05	00
00A8	4F	00	3A	D5	B7	06	08	81
00B0	10	FD	32	D5	B7	16	00	3A
00B8	06	00	5F	2A	D3	B7	06	08
00C0	19	10	FD	22	D3	B7	C1	E1
00C8	23	05	78	FE	00	C2	44	00
00D0	FD	E1	F1	E1	D1	C1	CD	18
00D8	F0	C9	00	00	08	D9	2A	D3
00E0	B7	22	09	00	3A	05	00	4F
00E8	3A	06	00	47	08	30	06	08
00F0	CD	03	F0	30	08	08	00	23
00F8	22	D3	B7	10	EF	2A	09	00
0100	22	D3	B7	3A	D5	B7	3D	32
0108	D5	B7	0D	79	FE	00	20	D8
0110	3A	05	00	47	3A	D5	B7	80
0118	32	D5	B7	2A	D3	B7	16	00
0120	3A	06	00	5F	19	22	D3	B7
0128	D9	08	C9	00	00	00	00	00

Bild 2 Großschrift für KC 85 (Hexdump)



Prozessor mit 16,5 MIPS

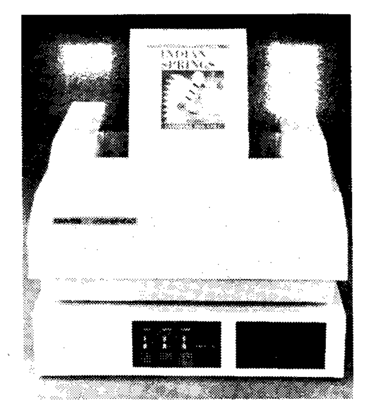
Der V80 von der Nippon Electric Corporation ist ein 32-Bit-Prozessor mit einer CISC-Architektur, der mit 16,5 MIPS (Millionen Instruktionen pro Sekunde) bei einer Taktfrequenz von 33 MHz rund 2,5mal schneller als der V70 ist (siehe auch MP 4/89, Seite 124). Die hohe Verarbeitungsleistung wird durch eine siebenstufige Pipeline-Architektur, je 1 KByte Cache-Speicher für Befehle und Daten sowie eine Fehlererkennung für Adreß- und Datenbus erreicht. Hergestellt wird der Chip in einer 0,8-μm-Technologie. Seine 930000 Transistoren befinden sich auf einer Chipfläche von 14,49 × 15,47 mm^2. Es existieren Versionen für Taktraten von 25 und 33 MHz. Ab 1990 soll auch eine 45-MHz-Version mit 22,5 MIPS verfügbar sein. *MP*

CADdy Junior

Für Einsteiger in den Bereich des computergestützten Zeichnens wird von Ziegler Instruments GmbH jetzt eine reduzierte Version des bekannten CAD-Paketes CADdy angeboten. CADdy Junior läuft auf IBM PCs und Kompatiblen und enthält Funktionen für das Zeichnen und Konstruieren mit geometrischen Grundelementen, für das Bemaßen, das Ändern von Geometrien und für die Symbol- und Ebenentechnik. Ein Austausch von Geometriedaten über eine DXF-Schnittstelle ist auch mit DTP-Programmen möglich. *MP*

PostScript-Farbdrucker

Von der Firma QMS wurde kürzlich der nach eigenen Angaben erste Farbdrucker vorgestellt, der die 35 Schriften und Farbsteuerkommandos der von Adobe entwickelten Programmiersprache PostScript enthält. Der ColorScript 100 arbeitet mit der Thermotransfer-Drucktechnik und liefert eine Auflösung von 300 × 300 Punkten pro Zoll. Als Farbträger dient eine Polyesterfolie, die mit den Grundfarben Cyan, Magenta, Gelb und Schwarz beschichtet ist. In vier Durchgängen erfolgt beim Druck das Schmelzen und Übertragen der Farben auf das Papier.
Damit sollen über 16 Millionen Farbkombinationen möglich sein. Der Drucker ist kompatibel zu den gängigen DTP-Programmen, die Farbe unterstützen und verarbeitet z. B. Dateien mit Farbinformationen des Ventura Publisher oder Adobe Illustrator.
Als Einstiegsmodell für Desktop Publishing-(DTP-)Anwendungen dient das Modell 20 für A4-Formate, mit einem 68020-Prozessor auf der Controllerkarte und 5 MByte RAM. Bei wachsendem Druckbedarf kann der Drucker zum Modell 30 erweitert werden – mit zusätzlich 4 MByte RAM, 20 MByte Festplatte und A3-Papierkassette. *MP*



Entwicklungstendenzen bei Bauelementen im kapitalistischen Markt

Der Markt für elektronische Bauelemente ist 1988 um mehr als 37 Prozent auf 46 Mrd. US-Dollar angewachsen. Dabei konnten die japanischen Hersteller ihre führende Position weiter festigen. Nach Angaben der Nachrichtenagentur Kyodo vereinigten die japanischen Hersteller 1988 rund 50 Prozent der gesamten Chipproduktion auf sich. Den Markt für Speicherchips soll Japan sogar zu 80 bis 90 Prozent beherrschen /1/.
Beachtliche Produktions- und Investitionspläne kündigten die führenden Elektronikkonzerne Nippon Electric Corp., Toshiba, Hitachi, Fujitsu, Mitsubishi Electric Corp. und Oki Electric Ind. an. Sie wollen ihre Halbleiterproduktionen in diesem Jahr um 15 bis 34 Prozent steigern. Danach soll ein Wert in Höhe von 2378 Mrd. Yen erreicht werden /2/.
Zu einem bedeutenden Faktor in der Halbleiterindustrie entwickelten sich die *anwendungsspezifischen ICs*. Ihr Anteil am internationalen Handel mit ICs soll 1990 rund 21 Prozent betragen. Der Absatz soll von 6 Mrd. US-Dollar 1987 auf 15 Mrd. US-Dollar 1992 ansteigen, wobei der Anteil der vollkundenspezifischen Schaltkreise an den Gesamtverkäufen von ASICs von 33 Prozent 1988 auf 15 Prozent 1992 zurückgehen soll. Im Gegensatz dazu soll der Anteil der Standardzellen-ICs von 16 Prozent 1986 auf 24 bis 25 Prozent anwachsen. Wesentlich an Bedeutung zunehmen sollen Gate Arrays, die 1992 mit 7,8 Mrd. US-Dollar mehr als die Hälfte des ASIC-Umsatzes auf sich vereinigen. In der Produktion von ASICs sind eine beginnende Spezialisierung auf bestimmte Schaltungstypen, die Schaffung von speziellen Design-Zentren sowie von Entwicklungssystemen für den Anwender erkennbar. Die wichtigsten Anwendungsgebiete für ASICs sind gegenwärtig die Telekommunikation (25 %), die Industrieelektronik und die Datenverarbeitung (jeweils 20 %) sowie der militärische Bereich. Zunehmende Bedeutung gewinnt die Kfz-Elektronik / 1/.
Obwohl sich *RISC-Prozessoren* nur unwesentlich von CISC-Prozessoren unterscheiden, ist ihre Rechengeschwindigkeit aufgrund eines verkleinerten Befehlssatzes rund 10mal schneller. Deshalb wird sich die RISC-Technologie künftig durch alle Bereiche der Computertechnik ziehen. Das Wachstum für RISC-Prozessoren im Computerbereich wird bis 1992 auf rund 60 Prozent jährlich veranschlagt. Am stärksten sollen die Verkäufe von Grafik-Supercomputern auf RISC-Basis zunehmen (um etwa 115 % jährlich). Als perspektivreich wird auch der Einsatz von RISC-Prozessoren in Systemen der Spracherkennung, der Laserdrucktechnik sowie der Automatisierungstechnik gesehen. Insgesamt wird für RISC-Prozessoren bis 1992 ein Wachstum von durchschnittlich 95 Prozent jährlich prognostiziert. Zur Zeit sind allerdings erst sehr wenige ausschließliche RISC-Systeme in Anwendung. In den meisten Fällen handelt es sich um hybride Lösungen, das heißt die Kombination beider Technologien /1/.
Der Markt für sehr schnelle Logik-Chips, der sich vor kurzem auf Großcomputer in Form von TTL beschränkte, hat sich bedeutend erweitert. Der Einsatzbereich für ICs mit sehr schnellen Verarbeitungszyklen weitet sich ständig aus. Die Produzenten von GaAs-Schaltkreisen haben Probleme wie hohe Herstellungskosten und relativ geringe Fertigungsausbeute zu lösen. Trotz dieser Einschränkungen sind GaAs-ICs immer dann von Bedeutung, wenn die Geschwindigkeit oder die Strahlenresistenz die wichtigste Rolle spielen. In allen anderen Anwendungsfällen behält der preisgünstigere Silizium-Chip weiterhin den Vorzug. Der Firma Gigabit Logic soll es zum Beispiel gelungen sein, die Fertigungsausbeute für GaAs im letzten Jahr von 30 auf 50 Prozent zu steigern /1/. Dataquest prognostiziert, daß sich der Markt für integrierte Schaltkreise auf GaAs-Basis bis 1991 im Durchschnitt um 96,2 Prozent auf rund 700 Mio US-Dollar ausweiten wird. Dabei setze sich verstärkt die Digitaltechnik durch. Der Verkauf von optoelektronischen Erzeugnissen auf GaAs-Basis soll sich 1991 von 2,1 auf 2,7 Mrd. US-Dollar erhöhen /3/.
Die wachsende Bedeutung von Sensoren und Sensorelementen kommt darin zum Ausdruck, daß es rund 10000 unterschiedliche Sensoren auf dem westlichen Markt gibt. Mit der Herstellung von Sensoren befassen sich gegenwärtig rund 2100 meist mittlere Betriebe, davon 1000 in Westeuropa (BRD: 500), 600 in den USA und 500 in Japan. Der Markt für Sensoren weist ein Wachstum von 15 bis 20 Prozent jährlich auf. Von den gegenwärtig produzierten Sensoren sind 20 Prozent Halbleitersensoren. Dieser Anteil soll bis 1992 auf 40 Prozent gesteigert werden. An Bedeutung zunehmen werden künftig faseroptische Sensoren und sogenannte Smart-Sensors, die elektronische Datenverarbeitung und den Prozeß des Messens in einem IC verbinden.
Zu den wichtigsten Trends bei den Sensoren gehören:
- Miniaturisierung
- Kombination von Sensoren mit Signalvorverarbeitung auf engstem Raum, d. h. intelligente Sensoren und Aktoren zu Sensorsytemen
- Entwicklung neuer Materialien für selektive Sensoren, vor allem für chemische und biologische Messungen.

Für den Markt von Leistungshalbleitern werden Wachstumsraten bis zu 50 Prozent jährlich vorausgesagt. Insgesamt wird die Leistungselektronik hauptsächlich von Leistungsdioden, bipolaren Leistungstransistoren und Thyristoren bestimmt. Bis 1992 soll dieser Anteil voraussichtlich auf 65 Prozent gesunken sein. Stark an Bedeutung zunehmen sollen die sogenannten intelligenten Leistungs-ICs in gemischten Technologien. Hauptanwendungsgebiete für Leistungshalbleiter im kommerziellen Bereich sind gegenwärtig die Industrieelektronik (33 %), die Kfz- und die Konsumgüterelektronik (14 %), die Computertechnik (11 %) und die Telekommunikation (9 %). Bis 1992 soll der Anteil der industriell eingesetzten Leistungshalbleiter auf 40 Prozent und der in Kraftfahrzeugen und anderen Konsumgütern auf 25 Prozent steigen /1/.
Die Technik der integrierten Halbleiterschaltungen ist eine Branche mit rascher Innovationsgeschwindigkeit und ebenso raschem wirtschaftlichen Wachstum. Die Technologie entwickelt sich heute schon schneller, als die Kunden sie dann in ihre Systeme integrieren können.
Als sehr bedeutsam wird für Halbleiter-Hersteller in Zukunft eine neue Technik zum Aufbau lokaler Verbindungen direkt auf dem Chip sein. Hierbei werden dünne Titan-Nitrid-Streifen erzeugt, die benachbarte Leiterbahnen miteinander koppeln und die ohne zusätzliche Komplizierung des Fertigungsgangs auf den Chip gebracht werden können.
Weitere Fortschritte auf dem Weg zu hochintegrierten Chips betreffen die Technik der vertikalen Kontaktierung, wo von gewöhnlich metallgefüllten Kontaktlöchern auf neuartige Wolfram-Kontakte übergegangen wird. Dabei steigt die Zuverlässigkeit, und man erhält glatte Oberflächen, was den Aufbau mehrlagig metallisierter Chip-Sandwiches erleichtere. Dreilagig metallisierte Chips machen ASICs, VLSI- und ECL-Schaltungen extrem hoher Leistung möglich /4/.
Elektromechanische und passive elektronische Bauelemente (Widerstände, Kondensatoren, Schalter, Steckverbinder, Leiterplatten) werden künftig nicht an Bedeutung verlieren und nach wie vor Wachstumsmärkte bleiben. Im Gegensatz zu den Halbleitern werden sie weniger von konjunkturellen Entwicklungen beeinflußt. Bei diesen Bauelementen wird langfristig mit jährlichen Steigerungsraten von durchschnittlich 5 bis 6 Prozent gerechnet /1/.

Quellen

/1/ Außenwirtschaft. – Berlin 17 (1989) 5. – S. 26, 27
/2/ Eildienst. – Berlin (1989) 3. – S. 4
/3/ Eildienst. – Berlin (1988) 253. – S. 4
/4/ Elektronik. München 37 (1988) 15. – S. 60–68

Fa.

Tastaturen

Die ständig steigende Komplexität und die Vielfalt von Standard- und Anwendersoftware für PCs stellen immer höhere Anforderungen an den Bediener. Für verschiedene Programmiersprachen, Textverarbeitung, Grafik sowie für Steuerungen müssen beispielsweise Sonderzeichen, Formeln, Kommandos oder Befehlssequenzen erzeugt werden; das Umschalten auf andere Zeichensätze (z. B. kyrillisch oder griechisch) ist erforderlich. Abhängig vom Programm oder von der Funktion ändert sich die Tastenbelegung. Die üblichen (IBM-kompatiblen) Tastaturen bekommen eine nahezu unüberschaubare Funktionsvielfalt. Hierbei sind 6fach-Tastenbelegungen keine Seltenheit mehr.

Das Problem mit einer größeren Tastatur lösen zu wollen führte dazu, daß Spezialtastaturen bereits bis zu 500 Tasten enthalten. Auch der Einsatz der Maus in Verbindung mit Bildschirmmenüs kann dieses Problem allein nicht lösen, da Eingabegeschwindigkeit und Bedienkomfort in vielen Fällen nicht ausreichend sind und die Fehlerrate höher als bei der Tastatur ist.

Die bei Standardtastaturen häufig vorhandenen 10 bis 12 Funktionstasten sind für viele Anwendungen ausreichend. Sie sind dann schwer bedienbar, wenn sie mehrfach belegt sind. Die Mehrfachbelegung der alphanumerischen Tasten ist ebenfalls bedienunfreundlich und bei einer Mehrfachbeschriftung der Tasten dazu noch unübersichtlich.

Die Forderung, die Tastaturen trotz unterschiedlicher oder mehrfacher Belegung leicht bedienbar zu machen, veranlaßte viele Hersteller, (XT-, AT- bzw. PS/2-kompatible) Tastaturen zu entwickeln, die diese Probleme (unterschiedlich) lösen. Allen Lösungen gemeinsam ist ein in der Tastatur integrierter Mikroprozessor (meist 8 Bit), 10 bis 34 Funktionstasten, bis zu 64 KByte RAM für das Speichern der momentanen Tastenbelegung (eventuell EEPROMs oder ein batteriegestützter SRAM) und bis zu 32 KByte EPROM für die Standardbelegung der Tastatur und für das Programm des Prozessors. Pro Funktionstaste können Ketten von bis zu 126 Zeichen gespeichert werden. Manche Tastaturen erlauben die freie Definition nahezu aller Tasten. Hierfür werden auswechselbare Tastenkappen oder Tastenschablonen für verschiedene Standardprogramme geliefert. Leere Schablonen ermöglichen die Beschriftung durch den Anwender. Für die Programmierung der Tasten kann die Tastatur ein spezielles LC-Tableau enthalten.

Weiterhin gibt es Tastaturen mit Anschlüssen für eine Maus oder einen Barcodeleser. Eingebaute Magnetkartenleser ermöglichen eine personengebundene Tastaturkonfiguration oder auch eine Benutzungsberechtigung. Für ein variables Absetzen der Tastatur werden Tastaturen batteriegepuffert und mit einer Infrarotfernbedienung versehen.

Zwei besonders interessante Lösungen sollen im folgenden etwas näher vorgestellt werden.

Bei der **Alphakey MF3** wird die Tastenbelegung mit einer Schablone kenntlich gemacht (Bild 1). Das Besondere an dieser Tastatur ist die Vielfalt der Programmierbarkeit der Tasten. Von den 119 Tasten können die Funktionstasten, die Sondertasten sowie der Kursor- und der Ziffernblock, also knapp die Hälfte aller Tasten, programmiert werden. Die Programmierung kann in vier Ebenen (normal und zusammen mit Shift, Ctrl, oder Alt) erfolgen. In der normalen oder PC-Ebene können 10 Zeichen pro Taste eingegeben werden. Bei den Funktionstasten (F1 bis F10) sind in zwei Ebenen auch 44 Zeichen möglich. Durch Schachtelung (eine Taste ruft eine andere) können die zehn Funktionstasten auch Sequenzen von bis zu 440 Zeichen erzeugen.

Die Programmierung ist denkbar einfach: Man drückt die Taste PRG und die zu belegende Taste und gibt dann die Sequenz ein; abgeschlossen wird wieder mit PRG. Soll die im RAM der Tastatur enthaltene Tastenprogrammierung nach Abschalten des Rechners erhalten bleiben, dann kann ein mit EEPROMs bestücktes *Memory*-Modul programmiert werden (EEPROM = elektrisch löschbarer, programmierbarer Nur-Lese-Speicher). Nach Wiedereinschalten des Rechners wird dann der *Memory*-Inhalt automatisch wieder in den RAM kopiert. Außerdem kann auch eine Datei auf Floppy- oder Harddisk erzeugt werden. Für die Anwender-Tastenbelegung gibt es Schablonen, die selbst beschriftet werden können.

Für Standardsoftware existieren bereits *Fix*-Module (mit EPROMs) und entsprechend beschriftete Schablonen (siehe Bild 1). Sollen zu einer Standardsoftware, die beispielsweise nur in einer Ebene Tasten definiert, zusätzlich (in anderen Ebenen) anwenderspezifische Tastenbelegungen programmiert werden, dann können auch *Kombi*-Module, die EPROMs *und* EEPROMs enthalten, eingesetzt werden. Der Modulwechsel kann bei eingeschaltetem Rechner erfolgen.

Vom Hersteller AFC Technology Leverkusen werden Versionen für PC-XT, /AT und PS/2 angeboten. Zusätzlich existiert eine Schnittstelle für Barcode-, Magnetkarten- und Klarschriftleser sowie für ein Spracherkennungssystem.

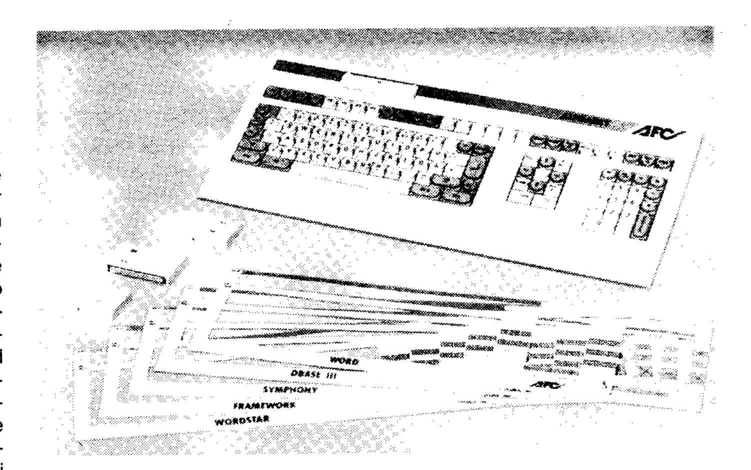


Bild 1 Alphakey MF3

Noch größere Variantenvielfalt und Bedienfreundlichkeit als die Alphakey bieten die verschiedenen Versionen von **The Board**. Die Tastatur besteht aus 117 oder 127 Tasten, von denen je nach Version 12 bis 113 Tasten mit einer LCD-Beschriftung versehen und damit auch programmierbar sind (Bild 2: 12 LCD-Tasten, Bild 3: 87 LCD-Tasten). Diese LCD-Tasten zeigen grafisch oder alphanumerisch die momentane Tastenbelegung an. Das ist für maximal 34 Funktionstasten mit 20 × 8 Punkten und für maximal 79 alphanumerische, Ziffernblock- und Kursorblocktasten mit 8 × 12 Punkten möglich.

Auf der Leipziger Frühjahrsmesse wurde die neue LCD-Taste The Key mit 32 × 16 Punkten angeboten, die wahlweise einen rot und/oder grün beleuchteten Hintergrund enthält.

Die Bildpunkte der LC-Displays können mit der mitgelieferten Software einzeln angesteuert werden. Die Darstellung ist positiv, negativ oder blinkend möglich. Die Programmierung der Tasten kann auch hier in vier Ebenen erfolgen (normal, Shift, Ctrl, Alt). In den 64 KByte großen RAM können für jede Tastenfunktion Ketten von bis zu 126 Zeichen eingegeben werden. Programmiert werden die Zeichenketten hier (ähnlich wie bei der Alphakey) mit der STRING-Taste. Diese Stringprogrammierung ist auch während eines Anwenderprogramms möglich, so daß häufig benötigte Textpassagen oder Steuersequenzen komfortabel gespeichert werden können. Ein Datenerhalt ist ausschließlich auf Diskette oder Festplatte möglich.

Jede Tastaturversion ist an XTs, an ATs oder an PS/2-Modelle anschließbar. Die Firma Hohe Electronics Neukirchen, bietet auch eine Version an, bei der zwischen PC-Tastaturbelegung und Terminalemulation für Großrechner (3270 oder VT220) umgeschaltet werden kann (siehe auch MP 12/1988, Seite 383). *MP-Hk*



Bild 2 The Board MF 1.12



Bild 3 The Board MF 1.24

Datenübertragungstreiber COCO

Der asynchrone Datenübertragungstreiber COCO stellt eine Erweiterung des Betriebssystems SCP dar und ermöglicht die Übertragung von Diskettendateien zwischen zwei über eine Standleitung verbundene Datenendeinrichtungen (DEE). Nach dem Laden verbleibt er im Hauptspeicher und schützt sich selbst vor dem Überschreiben durch andere Programme. Neben der Nutzung der SCP-Betriebssystemfunktionen ist die Lokalarbeit mit Standardsoftware (REDABAS, TP, POWER, KP) und unter Einhaltung bestimmter Randbedingungen mit Anwendersoftware (z. B. SCOM-LAN-Software) bei geladenem Treiber auch weiterhin möglich. Ein Dateitransfer wird bei Übereinstimmung einer vom Nutzer zu einem beliebigen Zeitpunkt eingegebenen Übertragungsstartzeit mit dem Stand der in COCO implementierten Echtzeituhr ausgelöst. Der Treiber gewährleistet dazu die permanente Anrufbarkeit der dateiempfangenden DEE. Durch eine spezielle Steuersoftware erfolgt die automatische Programmumschaltung vom Lokalarbeitsprogramm zum Datenübertragungstreiber COCO, das heißt, eine Datenübertragung unterbricht sowohl bei der dateisendenden als auch der dateiempfangenden DEE laufende Lokalarbeitsprogramme.
Nach Abschluß des Dateitransfers wird die Abarbeitung der zuvor unterbrochenen Lokalarbeitsprogramme fortgesetzt. COCO wurde bisher auf PC 1715 implementiert. Eine Version für BC 5120 mit E/A-Karte ATS 7028.20 ist vorgesehen.

VE BKK Senftenberg, Prozeßautomatisierung und Steuerung IRT, Brieske, 7803; Tel. 8 40 *Lieske*

Ein neues Datenbankkonzept

An der Hochschule für industrielle Formgestaltung Halle Burg Giebichenstein wurde ein Konzept für das Speichern von Fakten- und Bildinformationen entwickelt und verwirklicht. Die Basis dazu bildet die Kopplung eines Videorecorders mit einem BC 5120, die in Form einer Hard- und Software-Lösung geschaffen wurde (BUMA Karl-Marx-Stadt). Alphanumerisch beschreibbare Fakten werden mit dBase in Tabellenform erfaßt. Die informationelle Struktur des Gesamtsystems wird durch eine umfangreiche Menüsteuerung, die mit dem Programmierwerkzeug **MAKE-MENU** erzeugt wurde, realisiert.
Die Steuerung aller Vorgänge erfolgt mit einem in Turbo-Pascal geschriebenen Programm.
Die Bildinformationen werden als Videoclips auf Videoband gespeichert. Damit kann erreicht werden, daß hochwertige Bildinformationen (in TV-Qualität) in beliebiger Menge unabhängig vom Rechner gespeichert werden können. Das kurz beschriebene Konzept wurde an zwei Beispielen erprobt:
1. **ERGOFAKT** – ein Sachverhaltsspeicher zur Ergonomie
2. **PROKART** – eine Kartei für Vergleichsmusterdaten.

Hochschule für industrielle Formgestaltung Halle Burg Giebichenstein, Wissenschaftsbereich Designmethodik, Neuwerk 7, Halle, 4020; Tel. 85 03 73 *Prof. Dr. Frick*

Schaltungsanalyse und EPROM-Programmierung

Das erste Programm dient zur statischen logischen Analyse rein digitaler Schaltungen unter SCP. Es sind Schaltungsanordnungen mit bis zu 50 digitalen Grundelementen (alle Arten von Gattern und Flip-Flops) interaktiv testbar. Dieses Programm wurde unmittelbar zum Entwurf eines Gate-Array-Schaltkreises U 5200 ausgelegt und genutzt. Nutzen: Zeiteinsparung beim Entwurf von digitalen Schaltungen, insbesondere bei sequentiellen Automaten.
Die zweite Lösung bietet die Möglichkeit, EPROM-Typen von U 2716 bis U 2764 unter dem Betriebssystem OS/M auf dem P8000 programmieren zu können. Als Hardware wird der 8-Bit-P8000 und der EPROM-Programmer des P8000 vorausgesetzt.

VEB Meßgerätewerk „Erich Weinert“ Magdeburg, BfN, Straße der DSF 124, Magdeburg, 3010 *Dr. Engel*

Centronics-Schnittstelle am P8000

Oftmals ist die Anzahl der seriellen Schnittstellen am P8000 für den Anschluß von Terminals und anderen Geräten nicht ausreichend. Während einerseits eine serielle Schnittstelle durch einen Drucker belegt ist, wird andererseits in vielen Fällen die PIO-Schnittstelle, die für den EPROM-Programmer vorgesehen ist, nicht benötigt. Durch den Anschluß eines Druckers mit einer parallelen Schnittstelle über die PIO steht die serielle Druckerschnittstelle für andere Anwendungen zur Verfügung. Von diesen Überlegungen ausgehend wurde ein neuer Druckertreiber entwickelt, der es ermöglicht, Drucker mit Centronics-Interface unter dem Betriebssystem WEGA zu betreiben.

Zentrum für Forschung und Technologie, Kombinat VEB Elektro-Apparate-Werke Berlin-Treptow, Abt. WAE, Storkower Straße 101, Berlin, 1055; Tel. 4 38 85 09 *Gierth*

Pseudografiksoftware unter REDABAS

Das nachfolgend vorgestellte Programm ermöglicht es, mit Hilfe von REDABAS pseudografische Darstellungen auf dem Bildschirm bzw. auf dem Drucker erzeugen zu können.
Unser Programm läßt sich in vier Hauptabschnitte unterteilen
– Bereitstellen der Variablen und der Datenbankdatei
– Ausgabe der Koordinatenachsen, Beschriften, Legende
– Ausgabe der Säulengrafik auf dem Bildschirm
– Druckaufbereitung und Druck auf dem Nadeldrucker.
Durch die Nutzung pseudografischer Symbole ist eine höhere Genauigkeit der Bildauflösung und somit ein höherer Informationsgehalt als bei anderen Lösungsvorschlägen möglich. Zu Beginn der Bildschirmausgabe werden die Koordinatenachsen mit Beschriftung dargestellt. Die Legende für darzustellende Daten wird am rechten Bildschirmrand ausgegeben. Sie enthält in der derzeitigen Version folgende Angaben: den Kennbuchstaben der Säule, die Bezeichnung sowie den Wert.
Um die quasigrafischen Möglichkeiten des A 7100 nutzen zu können, werden verschiedene Zeichen im 2. Zeichensatz umdefiniert, so daß bei einer quasigrafischen Darstellung im Raster 24 × 80 ein entsprechendes Bild geschrieben wird.

Institut für Sozialistische Wirtschaftsführung, Forschungslabor, An der Schleuse 3/4, Woltersdorf, 1255 *Kasche/Standhardt*

Zugriff auf dBase-Dateien unter Turbo-Pascal 4.0

Die Interfaceprozeduren des Zugriffsmoduls db_LINK.TPU bilden eine satzorientierte Schnittstelle für die sequentielle oder wahlfreie Verarbeitung von Datenbankfiles, die mittels dBase II, III, III Plus oder REDABAS bzw. REDABAS-3 angelegt worden sind. Zusätzlich wird eine Unit db_SHELL.TPU angeboten, die den Funktionsumfang des Zugriffsmoduls auf ähnliche Routinen abbildet, wie sie für die Arbeit mit Textfiles bekannt sind. Diese Funktionen und Prozeduren können als Erweiterung von Pascal um einen Datenbankanschluß aufgefaßt werden.
Hier eine Auswahl neuer Sprachelemente: dbAssign, dbReset, dbRewrite, dbAppend, dbUpdate, dbSeek, dbSkip, dbLocateKey, dbLocateValue, dbGet, dbPut, dbRead, dbWrite, dbDelete, dbRecall.
Die Prozeduren dbGet und dbPut arbeiten mit einem vollständigen Datensatz. Die Konvertierung auswählbarer Felder in Variablen der Standard-Datentypen von Turbo-Pascal 4.0 erfolgt mit dbRead und dbWrite. Zur Demonstration der erreichten Datenunabhängigkeit wurde der dBase III Plus-Befehl LIST als Prozedur dbList bzw. dbStructure innerhalb der Shell simuliert. Ein Zugriff über Indexdateien ist nicht vorgesehen. Es können keine MEMO-Felder verarbeitet werden.
Die Units db_LINK.TPU und db_SHELL.TPU umfassen 27 bzw. 14 KByte. Ihre Verwendung ist an Turbo-Pascal 4.0 unter MS-DOS gebunden. Zur Nachnutzung werden der Quelltext der Shell, die beiden Units, acht Demonstrationsprogramme und eine Dokumentation angeboten.

Technische Universität „Otto von Guericke“ Magdeburg, Sektion Informatik, WB MI, PF 124, Magdeburg, 3010 *Herbst*

Grafik-Softwarepaket

Das hier vorgestellte System stellt eine Erweiterung und Anpassung des Systems GRAFIX-7100 für den A 7150 dar. Es ermöglicht eine Auflösung von 640 × 480 Pixeln.
Die Software ist für Turbo-Pascal geschrieben und wird als Unit **GRAF7150** in ein Pascalprogramm eingebunden. Sie wurde für wissenschaflich-technische Anwendungen konzipiert und ermöglicht im wesentlichen folgende Grafikfunktionen, die als Prozeduren im Nutzerprogramm aufgerufen werden können:
– grafische Primitive des GKS
– Splitting, 8 Farben (bzw. Graustufen), 4 verschiedene Schreibtypen
– umfangreiche Windowtechniken
– Weltkoordinatensysteme und Darstellungsprozeduren für die Grafik-Primitive und für reellwertige Funktionen in diesen WK
– Speichern von Grafikbefehlen und Windows auf Disketten
– zusätzliche Vektorzeichensätze für Textdarstellung
– interaktive Arbeit mit *Locator* bzw. *Stroke*
– Hardcopy auf Nadeldrucker
– Ansteuern des Plotters K 6418.
Es existiert ein analoges Softwarepaket für den A 7100 (als Include-Filesystem) unter Turbo-Pascal 3.0. Damit ist es möglich (mit gewissen Einschränkungen hinsichtlich der speziellen Windowoperationen), Programme auf beiden Rechnern kompatibel zu entwickeln.
Zusätzlich wurde ein umfangreiches Demonstrations- und Lehrprogramm erarbeitet, das alle Grafikprozduren menügesteuert vorstellt und so die Einarbeitung in das Programm sehr erleichtert. dabei erscheint auf dem Bildschirm neben der erzeugten Grafik jeweils der zugehörige Programmquelltext mit Erläuterungen.

Zentralinstitut für Physik der Erde, Bereich I, Telegrafenberg, Potsdam, 1560; Tel. 310/234 *Köhler*

Echtzeitentwicklungssystem für Signalprozessoren

Das Signalprozessor-Entwicklungssystem dient zum Entwickeln und Testen von Soft- und Hardware für den TMS 32020. Die Steuersoftware ist in C geschrieben und daher verhältnismäßig einfach auf andere Rechner übertragbar. Der eigentliche Entwicklungsmodul besteht aus einer DKL im Doppeleuropaformat, die über ein serielles oder ein paralleles Interface mit dem Steuerrechner verbunden ist. Neben dem Signalprozessor sind alle für das Betreiben des Systems erforderlichen Komponenten, wie 8 K × 16 Bit Programmspeicher und 16 K × 16 Bit Daten-RAM, sowie Bustreiber für alle nach außen gehenden Signale vorhanden. Der Nutzer betreibt seine Hardware an dem SP-Systembus und kann auf alle SP-Ressourcen nahezu uneingeschränkt zurückgreifen. So stehen für die Anwenderhardware u. a. sämtliche Interrupts, alle Ports, der interne Timer, die beiden UART-Kanäle und alle SP-internen RAM-Speicher (außer drei für die Monitorfunktion reservierte interne Datenwörter) zur Verfügung. Von den externen Programm- und Daten-RAMs stehen dem Anwender 7 K bzw. 15 K Wörter zur Verfügung. Darüber hinaus kann eine Speichererweiterung über den Systembus vom Nutzer durchgeführt werden. Der SP wird mit 20 MHz getaktet, und alle externen RAM-Zugriffe erfolgen ohne

Wait. Über den SP-Systembus kann der Anwender eine I/O-Emulation realisieren. Das bedeutet, daß alles bis auf die externen Speicher im Zielsystem emuliert werden kann.
Die Bediensoftware ist modular aufgebaut und hat einen Umfang von zirka 100 KByte. Das in Windowtechnik gestaltete Steuerprogramm ermöglicht eine übersichtliche und effektive Arbeit mit dem Entwicklungsmodul.
Zur Nachnutzung werden eine durchkontaktierte Modul-Leiterplatte, eine Interfacekarte, die Systemdokumentation mit Schaltplan, Bedienungsanleitung und Hardwarebeschreibung sowie die Software für die vom Nutzer gewünschten Steuerrechner bereitgestellt.

Akademie der Wissenschaften, Institut für Automatisierung, Rudower Chaussee 5, Berlin, 1199, Tel. 6740

Dr. Rompe

Diskettenmanipulierung mit SETDISK

Zum umkomplizierten Einstellen des Diskettenformates wurde **SETDISK** für den A 7100/50 unter SCP 1700 entwickelt. Die Parameterübergabe kann über die Kommandozeile sowie im Dialog erfolgen. Für die verbreitetsten 80-Spur-DS-Formate ist eine automatische Formaterkennung möglich.
Das Programm wurde in Turbo-Pascal geschrieben und liegt in zwei Versionen vor: die Normalversion für alle 80-Spur-Laufwerkstypen und die Version für den Laufwerkstyp SA460. Bei letzterer wird zusätzlich zum Einstellen des Diskettenformates die Stepperfrequenz auf 3 ms eingestellt. Dadurch wird die Diskettenarbeit um etwa 10 % beschleunigt und die lauten Geräusche des Steppermotors beseitigt.

VEB Funkwerk Berlin, Abt. MO, Wendenschloßstraße 142, Berlin, 1170; Tel. 6533607

Pawlowicz

Programmpaket für die Elektronikbranche

Wir haben ein Programmpaket für die Elektronikbranche entwickelt, das die üblichen Routinearbeiten in einem Handwerksbetrieb wie Rechnungslegung, Materialverwaltung, Lohnrechnung, Finanzbuchhaltung, Notizblock und Recherche vollständig rationalisiert. Die Programme sind in Turbo-Pascal unter CP/M 3.0 geschrieben und laufen bei uns bereits ein Jahr auf einem Schneider-Joyce PCW 8256 sowie einem Commodore 128 D.
Außerdem haben wir uns intensiv mit Kompatibilitätsproblemen und Hardwareanpassungen zwischen den genannten Rechnertypen und der äquivalenten DDR-Technik beschäftigt, wofür einige Stichworte stehen:
– Hardware-Kopplung Schneider – Commodore (V.24) zum Datenaustausch 5¼"-Floppy-Anpassung an Schneider-Joyce (720 KByte) mit Extern-Netzteil und schnellerem Diskettenzugriff (30 %)
– RAM-Erweiterung auf 512 KByte
– Fremdformateinstellung (Lesen, Schreiben, Formatieren) aller bekannten Diskettenformate einschließlich Datentransfer auch für DDR-Technik
– Implementierung bekannter Firmware (CP/A) sowie nützlicher Dienstprogramme (z. B. POWER) auf Commodore/Schneider/Amstrad
– Datentransfer CP/M – MS-DOS auf Diskette
– DDR-Drucker-Anpassung an Schneider-Joyce.
Alle Leistungen können durch uns auf der Basis gültiger Preiskarteiblätter nach Anfrage ausgeführt werden.

Digitalelektronik – Kleincomputer-/Taschenrechnerservice, Straße der DSF 9a, Cottbus, 7513; Tel. 523075

Manig

Kopplung SD 1156 – AC 7100/50

In unserem Betrieb wurde eine Hard- und Softwarelösung zum Anschluß eines Seriendruckers SD 1156 an den AC 71xx über die IFSP-Schnittstelle geschaffen.
Die Hardwarelösung beinhaltet eine Pegelanpassung zwischen dem Computer und dem Drucker. An beiden Geräten sind keine technischen Änderungen vorzunehmen.
Softwaremäßig wurde folgendes realisiert:
– Gerätetreiber für DCP 1700 (Kanal LPTx)
– SCP 1700 im Betriebssystem integriert (steuerbar über das I/O-Byte)
– Vor- und Rückwärtsdruck
– Fehlerbehandlung
– Ausblenden von ESC-Steuerfolgen
– Druckertest analog GTP 4200.

VEB Papierfabrik Dreiwerden, Hauptstraße 5, Abteilung ORZ, Schönborn-Dreiwerden, 9256; Tel. 5121 HA 335

Schultes

Shell-Menü-Hilfe für P8000/WEGA

Für P8000/WEGA-Betreiber bieten wir eine komplexe *Shell-Menü-Hilfe* zur Nachnutzung an. Mit dem durchgängigen Menüsystem können auf einfache und bequeme Art fast alle ständig wiederkehrenden Arbeiten eines Superusers ausgeführt werden. Die Anwendung bietet dem Nutzer eine bequem handhabbare Benutzeroberfläche zum WEGA-Betriebssystem mit seinen vielfältigen und zahlreichen Werkzeugen an Grundsoftware. Es handelt sich um eine komplexe, integrierte Lösung, die zirka 150 Menüpunkte gestaffelt in 3 Ebenen zur Abarbeitung anbietet. Zusätzlich sind auch Lösungen wie Notizbuch, Adreßbuch, Terminkalender, Systemabrechnung, Virustest und anderes eingearbeitet.
Die Lösung liegt als C-Shell-Skript vor, und für eigene Erweiterungen/Ergänzungen sind die notwendigen Vorkehrungen getroffen. Zur einfachen und sicheren Übernahme wird ein entsprechendes Installationsprogramm mitgeliefert.

VEB Datenverarbeitungszentrum Cottbus, Bereich Absatz, Wilhelm-Pieck-Straße 5, Cottbus, 7500; Tel. 484209

Röder

DCP-Drucktreiber für SD 1152

Im Seehafen Rostock wurde ein Druckerdriver zum Anschluß eines Seriendruckers SD 1152 mit IFSS-Anschluß an einen A 7150 unter DCP Version 3.2 entwickelt und seit längerer Zeit erfolgreich genutzt.

VEB Seehafen Rostock, Abt. LE, Rostock – ÜSH, 2540; Tel. 3663124

Schmeling

Neue RELA-Versionen 3.0 und 4.0

Gegenüber der Vorgängerversion 1.4/T von RELA für den KC 85/3 (siehe MP 4/1989, Seite 122) ergeben sich für die **Version 3.0/T** folgende Veränderungen:
– Kaltstart/CAOS wird mit OPEN n (statt RELA n) erreicht.
– Bei Funktions- oder Feldvorwahl wird die Signalkennzeichnung erkannt.
– Bei den Kommandos ALOAD, ASAVE, SOND-SUCH entstehen keine Überlappungen der Fenster.
– Beim Kommando SOND-SUCH wird Dateinummer angezeigt. Anwenderprogramme und Dateien sind zu anderen Versionen kompatibel.
Die RELA-**Version 4.0** für den KC 85/4 hat annähernd den gleichen Leistungsumfang wie die Version 3.0. Sie ermöglicht störungsfreie Bildschirmzugriffe und hat einen verbesserten Zeichensatz. Dem Anwender stehen die Erweiterungen der 3. Tastaturebene zur Verfügung, und es werden die Treiberroutinen des erweiterten Betriebssystems genutzt. Als Standard sind die Drucker K 6313/20 oder Kompatible vorgesehen.

ELECTRONIC-SERVICE Helmut Ulbricht, Göttiner Landstraße 7, Brandenburg, 1800; Tel. 522076

Zierott

Kopplung von MC 80.3x und PC

Für die On-line-Kopplung des Rechners MC 80.3x mit Personalcomputern der Typen PC 1715, AC 7100, AC 7150, EC 1834 sowie IBM-Kompatiblen wurde eine prinzipielle Softwarelösung auf der Basis der jeweiligen seriellen Schnittstellen entwickelt. Verfügbar ist derzeit die Lösung für den A 7100 auf der Basis der IFSS-Schnittstelle der ASP K 8071 mit einer effektiven Übertragungsrate von 20 kByte/min. Im Master-Slave-Betrieb wird die Übertragung von beliebigen Speicherbereichen des MC 80 zum Diskettenlaufwerk des PCs und zurück in Form von Blöcken zu je 256 Byte im Polling-Betrieb durchgeführt. Für den MC 80 (Slave) wird ein entsprechendes Treiberprogramm bereitgestellt, das auf einen EPROM zu übertragen ist. Alle wesentlichen Manipulationen werden im Dialogbetrieb vom PC-Master aus organisiert. (Betriebsprogramm MCPC in Turbo-PASCAL V.3 für die jeweils zum Einsatz gelangenden PCs.)

Institut für Polymerenchemie der ADW der DDR, Abt. Elektronik/Rechentechnik, Kantstraße 55, Teltow, 1530; Tel. 46264

Dr. Fanter/Urban

Basic-Vokabeln über Tastatur verfügbar

Als kleine Unterstützung für den Basic-Programmierer auf dem KC 85/3 entstand eine kleine Maschinenroutine, die nach Anwahl des Befehls über die Tastatur die Vokabel an den Basic-Interpreter **übergibt**. Die Befehle sind mehrmals hintereinander verwendbar, weil sie auf F1 gespeichert werden. Die Dokumentation und die Befehlsliste werden bei Zusendung einer Kassette sowie von Rückporto kostenlos abgegeben.
Das Programm darf an Dritte weitergegeben werden, wenn die Urheberschaft nicht verändert oder verschleiert wird.

Uwe Stieber, Erich-Jungmann-Straße 8, Radebeul 1, 8122

MINICALC

Auf Grund des regen Interesses und der vielfältigen Nachfragen teilen wir mit, daß das in der MP 12/88, Seite 376, angebotene Tabellenkalkulationsprogramm **MINICALC** nicht nur auf dem KC 85/3, sondern auch auf den Rechnern KC 85/2, KC 85/4 und KC 87 lauffähig ist.

VEB Cottana Mühlhausen, ORZ, Johannisstraße 44, Mühlhausen, 5700

Ulbrich

Kopplung KC 85/3 – PC 1715

Zur weiteren Erhöhung der Effektivität des Programmpaketes *Hydraulik/Elektro* (siehe MP 4/1988, Seite 109) wurde eine Software zur Kopplung von KC 85/3 und PC 1715 im Kombinat Baukema entwickelt.
Damit ergeben sich folgende Vorteile bei der Nutzung des KC 85 als Grafikentwicklungssystem:
• Nutzung des Massenspeicher- und Datenverwaltungssystems des PCs
• Reduzierung der zeitaufwändigen Speichervorgänge von Programmen und Daten des KCs
• Erhöhung der Datensicherheit bei Datentransfer
• Nutzung der bekannten Dienstprogramme auch für KC-Files
• Reduzierung des Zeitaufwandes bei Datenübertragung um 50 %
• Veränderungen bzw. Erweiterungen der Rechner-Hardware sind nicht erforderlich; es werden die geräteeigenen V.24-Schnittstellen genutzt.

VEB Kombinat Baukema, Abt. TVRK, Gerhard-Ellrodt-Straße 21, Leipzig, 7034, Tel. 7925278 oder 314

Walter/Lehmann

Neues OS für COMP JU+TE R

Von uns wurde ein Betriebssystem für den in der Zeitschrift Jugend und Technik vorgestellten Computer entwickelt. Das Programm ist nur 2 KByte lang und bietet folgende Möglichkeiten:
– Bildschirm: 128 × 128 Punkte, 16 × 16 Zeichen
– Bildschirmeditor (damit Korrektur eingegebener Zeilen möglich)
– Maschinensprachmonitor (Hexlisten, -schreiben usw.)
– Kassettenaufzeichnung (auch von Maschinenprogrammen) mit Fehlerkontrolle
– Ansteuerung beliebiger Drucker vorgesehen
– echte Shift-Taste.

Computerclub der SPS „Heinrich Hertz", Frankfurter Allee 14a, Berlin, 1035

Lüdtke

Computerwissen für alle

von E. Schiller, VEB Fachbuchverlag Leipzig, 1988, 232 Seiten, 111 Illustrationen, DDR 12,– M

In interessanter und beeindruckender Weise wird in diesem Buch, das eine Zwischenstellung zwischen einem Fachbuch und einem populärwissenschaftlichen Buch einnimmt, die gesamte Breite der Informatik von der Logik und den mikroelektronischen Grundlagen über die Programmiersprachen und Betriebssysteme bis hin zur Anwendung der Computer in allen gesellschaftlichen Sphären dargestellt.
Es ist so geschrieben, daß es für jeden Leser verständlich sein dürfte, jedoch auf Grund der Breite auch für viele Spezialisten noch Wissenswertes enthält. Die Verständlichkeit des dargelegten Fachwissens wird sehr wirkungsvoll durch die hervorragenden Illustrationen von Wolfgang Parschau unterstützt.
Natürlich sind bei der Breite des Buches kleinere Unexaktheiten nicht ausgeschlossen, auf die hier auch nicht näher eingegangen werden sollte. Bei der Lektüre ist mir aber ein prinzipieller Mangel aufgefallen, der in einer undialektischen Betrachtungsweise historisch bedingter Entwicklungen in der Computertechnik und Informatik besteht. Es ist richtig, daß in der Anatomie „digit" Finger heißt, aber in der Mathematik bedeutete „digit" schon immer Ziffer. Ein Digitalrechner ist damit ein Ziffernrechner, im Gegensatz zum Analogrechner (S. 14).
Eine Folge von 8 Bit nennt man nicht „bei den meisten Computern" Byte, sondern ist seit Ende der sechziger Jahre das Byte (S. 17).
Es ist auch nicht richtig, das Konzept des Harvard-Rechners Mark 1 von Howard Aiken aus dem Jahre 1944 als Alternative zum Von-Neumann-Prinzip wieder aufleben zu lassen. Auch wenn bei modernen Ein-Chip-Mikrorechnern (wie auch manchen Kleinrechnern) eine Trennung von Daten- und Befehlsadreßraum aus unterschiedlichen Gründen vorgenommen wurde, darf man dieses Prinzip nicht mechanisch auf eine Stufe mit dem getrennten Daten- und Programmspeicher des Harvard-Rechners stellen (S. 27). Man kann auch nicht einverstanden sein, daß als Ausgangspunkt der Kleinrechnerentwicklung der Einsatz in kleineren Betrieben stand. Der Ausgangspunkt war, daß Anfang der sechziger Jahre ein realer Bedarf an Rechnern für Aufgaben der Produktionsautomatisierung, also interaktiv einsetzbarer Rechner bestand. Dafür aber waren die Großrechner nicht geeignet, die für den zentralen Einsatz in Rechenzentren projektiert waren. Für die Kleinrechnerentwicklung war in erster Linie das neue Einsatzgebiet von ausschlaggebender Bedeutung. Somit wurde die Kleinrechentechnik auch der Wegbereiter der Mikrorechentechnik (S. 45 ff).
Dennoch war für mich diese Lektüre sehr interessant, so daß ich das Buch jedem interessierten Leser nur empfehlen kann. *Prof. Dr. Th. Horn*

Zeitschriftenumschau

E. M. Projdakov:
Datengesteuerter Disassembler
Mikroprocessornyje sredstra i sistemy 5 (1988) 5

Die meisten in der Literatur veröffentlichten Arbeiten zur Disassemblierung befassen sich mit dem trivialen Problem des inkrementellen Disassemblers. Dieser Artikel geht darüber hinaus.
Eingangs wird die Notwendigkeit von Disassemblern oder Reassemblern insbesondere für Ziele der Adaption begründet. Es werden Ein- und Mehrpaßdisassembler unterschieden und deren potentielle Übersetzungsmöglichkeiten angegeben.
Als Grundaufgabe der Disassemblierung wird die Trennung von Befehlskode- und Datenbereichen angesehen. Dabei wird erkannt, daß diese Aufgabe nichttrivialer Natur ist. Als Nebeneffekt der Disassemblierung fällt das Erkennen unbenutzter Programmzweige an. Auf grundsätzliche Schwierigkeiten bei der Disassemblierung, die durch Befehlskodemanipulation entstehen, wird nur kurz eingegangen. Bis hierhin geht der Artikel nicht wesentlich über die sehr wenigen bekannten Arbeiten hinaus.
Sein wesentlicher Beitrag besteht in Überlegungen zur rechnergestützten Generierung von Disassemblern für eine ganze Klasse von Mikroprozessoren. Dabei wird ein universelles Grundprogramm benutzt, dessen Ausführung von einer Tabelle gesteuert wird. Diese Tabelle enthält alle Informationen über den Befehlssatz, insbesondere zum Befehlsformat in Form von Schablonen. Der Disassembler enthält noch einige Editorroutinen, die das Beschreiben der Steuertabelle erleichtern. Die Generierung eines Disassemblers für den Befehlssatz von 280 erfordert nur einige Stunden Arbeitszeit. Der Disassembler ist in einer höheren Programmiersprache geschrieben und läuft unter den Betriebssystemen CP/M 80, ISIS-11, Mikro-DOS, BOS 1810, MS-DOS, CP/M86 und den ihnen kompatiblen.

A. N. Archangelskij, A. A. Orechov:
Multiprogrammierung in der Sprache C
Mikroprocessornyje sredstra i sistemy 5 (1988) 3 S. 41–42

Gegenwärtig gibt es die Tendenz, Mittel zur Multiprogrammierung nicht in die Programmiersprache selbst einzubinden (Modula-2), sondern einen Exekutionskern in Form einer Bibliothek von Systemprogrammen zu schaffen. Die Autoren beschreiben einen von ihnen entwickelten Realzeitexekutionskern, der für die Anwendung in Prozeßrechnern gedacht ist, die dem Befehlssatz nach mit der Elektronika 60 kompatibel sind. Der Exekutionskern ist für die Arbeit im Programmiersystem DECUS C bestimmt; er ermöglicht es, Parallelprozesse in der Sprache C zu programmieren. Insbesondere ist er geeignet für Systeme mit einer großen Anzahl von zyklischen Prozessen und kurzen aktiven Zeitabschnitten. Das Hauptobjekt, mit dem der Exekutionskern operiert, ist die Task. Für jede Task wird ein Taskdeskriptor erzeugt, der den Kontext der Task enthält. Zur Wechselwirkung und Synchronisation der Prozesse werden Signale benutzt analog den Signalen, wie sie von Modula-2 bekannt sind. Im C-Quelltext wird jede Task als Funktion vom Typ „Prozeß" mit einer beliebigen Anzahl von Argumenten beschrieben. Das erste Argument jeder Task muß vom Typ Deskriptor sein. Während der Programmabarbeitung können mehrere unabhängige Aktivierungen einer Task generiert werden, im allgemeinen mit verschiedenen Prioritäten und Argumenten. Dabei ist zu beachten, daß die globalen Variablen und die lokalen statischen Variablen gemeinsam für alle Aktivierungen einer Task sind.
Der Exekutionskern ist in Makroassembler geschrieben und wird an das compilierte C-Programm angebunden. Im gebundenen Programm ist der Umfang des Kerns geringer als 250 Worte. Der Artikel enthält eine verbale Beschreibung der Funktionen des Exekutionskerns und der benutzten Systemprogramme.
Weiterhin wird ein kurzes Programmbeispiel zur Anwendung angeführt und eine Liste von Bedingungen für die C-Programme, die den Exekutionskern benutzen.

V. A. Serebrjakov:
Attributierte Translation logischer Ausdrücke
Programmirovanije 14 (1988) 1 S. 7–11

In dem Artikel werden Fragen der Compilation von logischen Ausdrücken der Sprache Modula-2 auf der Basis attributierter Grammatiken betrachtet. Attributierte Grammatiken sind ein modernes, effektives Spezifikationsmittel zur Beschreibung von Übersetzungen. Da sie eine syntaxorientierte Beschreibung der Semantik darstellen, kombinieren sie in vorteilhafter Weise Anschaulichkeit der Beschreibung mit strengem Formalismus, der die Grundlage für eine automatische Generierung des Übersetzers ist. Auf dem Gebiet der Compiler-Compiler haben in den letzten Jahren die attributierten Grammatiken eine führende Stellung eingenommen. Aber auch für den datenstrukturorientierten Softwareentwurf finden die attributierten Grammatiken als Basisformalismus zunehmend Beachtung. Der Autor beschreibt die konkreten Besonderheiten der Kodegenerierung für logische Ausdrücke auf der Basis des Compiler-Compilers *Super* für den Minirechner SM 4. Aus der Definition von Modula-2 folgt die Realisierung der logischen Ausdrücke durch eine äquivalente Steuerflußstruktur auf der Basis von Test- und Verzweigungsbefehlen. So werden die Verzweigungsbefehle „beg" und „bne" zur Realisierung von Konjunktion und Disjunktion benutzt. Dabei ist keine eindeutige Zuordnung vorhanden, das heißt, ob „beg" z. B. für eine Konjunktion oder Disjunktion benutzt wird, hängt von der Stellung des Befehls zu anderen Befehlen ab. Die Information darüber kann durch Attribute übermittelt werden. Die Kodegenerierung erfolgt von einem Zwischenkode „Leader" aus, der ein attributierter Syntaxbaum ist.
Der Kodegenerator wird durch eine attributierte Grammatik beschrieben, die Eingangssprache für den Compiler-Compiler „Super" ist. Diese Grammatik muß L-attributiert sein, das heißt die Berechnung aller Attributwerte bei einer einmaligen Traversierung des Baumes von oben nach unten und von links nach rechts gewährleisten.
In dem Artikel sind die Details der entworfenen attributierten Grammatik beschrieben: Auswahl der Verzweigungsbefehle, Bestimmung der Marken, Generierung der Sprungbefehle. Für ein Beispiel eines komplizierten logischen Ausdruckes ist der attributierte Syntaxbaum angegeben.
Zusammengestellt von Dr. B. Stiefel

Lehrmaterial für 16-Bit-Arbeitsplatzcomputer

Aufbauend auf unserem Schulungsmaterial zum PC-Grundkurs (siehe MP 1 (1987) 5, S. 153) wurde vom CAD/CAM-Zentrum unserer Einrichtung ein vierteiliges Lehrmaterial für die Benutzung der 16-Bit-Gerätetechnik der DDR (A 7100, A 7150, EC 1834, P 8000) erarbeitet. Folgende Teile liegen bisher als Diskettendatei mit einem Umfang von je 15 Seiten zur Nachnutzung vor:
Teil 1 Der Übergang vom SCP zum DCP
Teil 2 DCP-Optionen und Befehle
Teil 3 Das Betriebssystem SCP für A 7100
Teil 4 Das Betriebssystem WEGA für P 8000.
Senden Sie bitte Ihre Nachnutzungswünsche mit einer 5,25-Zoll-Diskette an folgende Kontaktadresse oder rufen Sie uns an (Tel. 59 23 65).

Technische Universität Magdeburg, CAD/CAM-Zentrum, Postfach 124, Magdeburg, 3010 *Dr. Springer*


Literatur

Taschenbuch Elektrotechnik
Band 3: Bauelemente und Bausteine der Informationstechnik

Herausgegeben von Prof. Dr. sc. techn. Dr. techn. h. c. Eugen Philippow. 3., stark bearbeitete Auflage. Zwei Teile. 1192 Seiten, 1525 Bilder, 400 Tafeln, Kunstleder, im Schuber, Teile I/II 44,– M, Ausland 64,– DM. Bestellangaben: 553 713 5/Tb. Elektro 3
Die Auflage wurde völlig überarbeitet und auf den neuesten Stand gebracht.



Bericht



'89er Messe-Leitthema: Flexible Automatisierung

Die Integration von mikroelektronischen Baugruppen und Geräten in Anlagen des Werkzeug- und Verarbeitungsmaschinenbaus gilt weltweit als Erfolgsrezept für höhere Produktivität und Erzeugnisqualität. Sie ist die Voraussetzung für den Übergang von der Einzelmaschine zu flexiblen automatisierten Maschinensystemen. Das Leitthema der beiden Leipziger Messen 1989 *Flexible Automatisierung* stellt sich das Ziel, diese enge Verflechtung von Mikroelektronik und Maschinenbau im Exponateprogramm der in- und ausländischen Aussteller wie auch in wissenschaftlich-technischen Veranstaltungen darzustellen und zukunftsorientierte Lösungen aufzuzeigen. Vor allem in der metallverarbeitenden Industrie, aber auch in anderen Bereichen, ist die flexible Produktionsautomatisierung das derzeitig progressivste Fertigungskonzept. Mit diesem Leitthema setzt die Leipziger Messe den seit Jahren erfolgreich praktizierten Weg fort, international aktuelle ökonomische Entwicklungen in den Mittelpunkt des Messegeschehens zu rücken. Die große Nachfrage nach Ausstellungsfläche aus dem Ausland, insbesondere für den Branchenkomplex Werkzeugmaschinen und Werkzeuge, sowie die zahlreichen Anmeldungen für Fachvorträge zum Leitthema waren bei der Frühjahrsmesse Ausdruck für das rege Interesse an dieser Thematik. Der Einfluß der Informationstechnik auf die Produktion zeigte sich natürlich nicht nur in der Verflechtung mit dem Maschinenbau – demonstriert beispielsweise in Halle 20 –, sondern in praktisch allen Fertigungsbereichen.

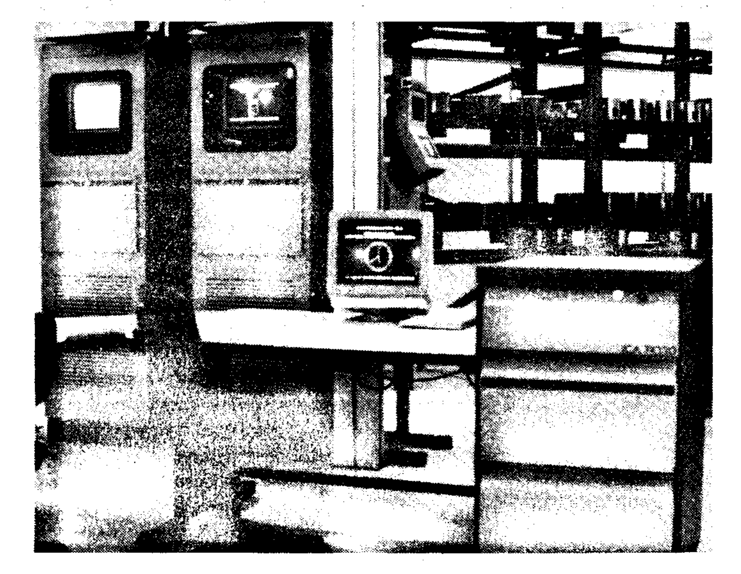
Wir haben uns im ersten Teil unseres Berichtes von der LFM vor allem auf das Gebiet des rechnergestützten Entwurfs konzentriert, wobei es nicht mehr zweckmäßig erschien, hierbei eine strikte Trennung in Hardware und Software vorzunehmen. Daran anschließend widmen wir uns der Computertechnik, die als nicht CAD/CAM-spezifisch einzuordnen ist, beispielsweise Personalcomputer.

In der Fortsetzung des Messeberichtes in MP 8/89 werden wir Sie über periphere Hardware, Bauelemente sowie weitere Softwareangebote informieren.

CAD/CAM/CAP

Das Spitzenerzeugnis des Kombinates Automatisierungsanlagenbau für die flexible Automatisierung war der Industriecomputer **ICA 700** (Industrie-Computer-Automatisierung). Erstmals wurde die Terminalvariante des ICA 700 als Leitrechner für eine Zelle eines flexiblen Fertigungssystems für Dreh- und Gehäuseteile mit Hochregallager und mit Bestükkungsroboter vorgestellt (siehe Bild rechts). Das Einsatzspektrum dieser Industriecomputerfamilie reicht von der Programmierung unter DCP über den Einsatz für die Meßwerterfassung und -bearbeitung, für das Steuern und Überwachen von Aggregaten, Teilanlagen und Fertigungsstraßen sowie für die Laborautomatisierung bis hin zu Inbetriebnahme und Service von Automatisierungsgeräten und -anlagen. Erreicht wird diese Universalität durch ein Zweirechnerkonzept (PC und Echtzeitrechner) und eine modulare Baugruppenstruktur des Echtzeitcomputerteils sowie durch eine Gerätefamilie. Diese Gerätefamilie besteht aus der modularen, durch den Anwender konfigurierbaren Variante ICA 710.10, der Schrankvariante ICA 710.20 (im Bild im Hintergrund) sowie der Terminalvariante ICA 710.30 (im Bild im Vordergrund). Die Geräte der ICA 710-Reihe verwenden den 8086-kompatiblen Prozessor K 1810 WM86, und ihr PC-Teil ist kompatibel zum PC/XT. Der PC-Teil der künftigen Gerätereihe ICA 720 dagegen wird PC/AT-kompatibel sein und um die mobile Variante ICA 720.40 ergänzt werden.

Am Beispiel des ICA 710.20 sollen nachfolgend einige Hauptmerkmale erwähnt werden: Der vorrangig für den Einsatz unter rauhen Bedingungen konzipierte Industriecomputer in Schrankausführung besteht aus einem Hauptrechner (PC-Teil) und einer Prozeßkoppeleinheit (Echtzeitcomputersystem). Beide Rechnerteile sind über ein internes ICA-Interface gekoppelt. Der PC-Teil enthält eine Harddisk mit 20 bis 40 MByte, zwei Floppylaufwerke mit je 720 KByte, 256 KByte RAM (auf 640 KByte erweiterbar), 32 KByte ROM und den mit 4,915 MHz getakteten K 1810 WM86 (optional K 1810 WM87). Als Betriebssystem dienen DCP oder das UNIX-kompatible MUTOS. Der Echtzeitcomputerteil besitzt einen Systembus und einen Residentbus. Der multimasterfähige Systembus ist kompatibel zum Multibus, zum AMS-Bus sowie zum Systembus des MMS 16 (maximal 16 Master). Der Residentbus ist bis auf die fehlende Multimasterfähigkeit mit dem Systembus identisch; alle Operationen laufen dadurch konfliktfrei und mit maximaler Geschwindigkeit ab. Der Echtzeitcomputerteil enthält eine Zentralbaugruppe mit K 1810 WM86 und K 1810 WM87, neben DRAMs und EPROMs einen batteriegestützten SRAM, weiterhin die Schnittstellen: IFSS, IFSP, Centronics, V.24, IFLS und optional das ethernet-kompatible ROLANET 1. Für die Prozeßeingabe und -ausgabe stehen sechs verschiedene digitale und zwei verschiedene analoge Baugruppen zur Verfügung; maximal 22 PEA-Baugruppen sind möglich. Als Betriebssysteme können BOS K 1810 und das EMOS 2 genutzt werden.



Minispot ist eine Variante des Prozeßrechnersystems SPOT 83 aus den rumänischen ICE-Felix-Werken. Das System zeichnet sich durch seine Variabilität sowohl bezüglich der CPU – die einen Single Board Computer darstellt – als auch der prozeßspezifischen Module aus. Während die Version CP 04 beispielsweise den 8-Bit-Prozessor 8080 und die Version CP 05 den 8-Bit-Prozessor Z 80 enthält, wurde in Leipzig die Variante CP 06 mit dem 16-Bit-Prozessor 8086 vorgestellt (Bild 1, alle Farbbilder auf den Umschlagseiten 2 bis 4). Das Board enthält außerdem bis zu 128 KByte RAM und 32, 64 oder 128 KByte ROM. Als Schnittstellen können 4 × RS 232C oder optional die 20-mA-Stromschleife genutzt werden. Somit ist es möglich, den Rechner Minispot als Stand-alone-System, als Local-System oder als Remote-System einzusetzen; zum Beispiel in Verbindung mit den rumänischen Minirechnern CORAL oder Independent I 102F/I 106, welche PDP-11-kompatible Software nutzen.

Die bulgarische Exposition stand in diesem Jahr weniger im Zeichen der Hardware als – dem Motto der Messe entsprechend – vielmehr im Zeichen der Produktionsorganisation. So waren zwar Bauelemente und Speichertechnik zu sehen, vor allem aber Vernetzungsmöglichkeiten der Computer sowie CAD/CAM-Softwarelösungen. Beispielsweise ein Programmsystem zur konstruktiven und technologischen Vorbereitung der Produktion am Terminal EC 8531.M2 und ein programmtechnischer Komplex zur Automatisierung diskreter und diskontinuierlicher Prozesse mit Namen PROCON.

Eine Voraussetzung für die effektive Automatisierung des Produktionsprozesses ist die schnelle, zuverlässige und kostengünstige Entwicklung von elektronischen Steuerungen. Dabei wird in den nächsten Jahren der Anteil von anwenderspezifischen Schaltkreisen (ASICs), die ganze Leiterplatten ersetzen können, sprunghaft ansteigen. Hierfür müssen leistungsfähige Entwurfssysteme zur Verfügung gestellt werden. Damit sind CAD-Systeme die Voraussetzung für die Entwicklung wieder neuer CAD/CAM-Systeme.

Ein Hauptvertreter unter den ASICs ist der Gate-Array-Schaltkreis. Für den Entwurf des Gate-Array-Systems U 5200/5300 bot der VEB Forschungszentrum Mikroelektronik Dresden das Programmsystem **ARCHIMEDES** an (Bild 2). Ein effektiver Schaltungsentwurf mit kurzen Entwicklungszeiten und bei einer hohen Entwurfssicherheit ist damit im Dialog und in durchgängiger Systemnutzung möglich. Ständige Statusinformationen über den gegenwärtigen Bearbeitungsstand und Hilfestellungen sowie eine Vielzahl auf richtige Reihenfolge kontrollierter Prüfschritte gehören ebenfalls zu diesem CAD-System.

Die Eingabe von Textzeilen in einer speziellen Netzbeschreibungssprache ist die Voraussetzung für alle weiteren Schritte, vom Auflösen des Textes und dem Erzeugen eines Netzwerkcodes über die statische Logiksimulation, die Plazierung, die Grobtrassierung, die dynamische Logiksimulation bis hin zur automatischen Testsatzgenerierung.

In einer integrierten Macro-Bibliothek

sind bereits fertig entwickelte Schaltungsgrundelemente (physische Macros) und ihre logischen, elektrischen, dynamischen und topologischen Beschreibungen enthalten und dokumentiert. Darüber hinaus werden komplexe Schaltungen, die aus den Schaltungselementen zusammengesetzt sind, wie Zähler, Schreibregister oder arithmetische Einheiten, als sogenannte Software-Macros angeboten, wobei die Plazierung der einzelnen Gatter jedoch erst im Rahmen der Plazierung des gesamten Schaltkreises erfolgt. ARCHIMEDES ist auf 32-Bit-Rechentechnik (z. B. K 1840) lauffähig.

Als neues interaktives grafisches Terminal für die Reihe der 32-Bit-Rechner des Kombinates Robotron war das **IGT K 8919.11** ausgestellt (Bild 3). Es beinhaltet einen Grafikprozessor, der einen Bildspeicher mit 8 Speicherebenen zu je 1280 × 1024 Bit (≙ Bildpunkten) verwaltet. Für die Kopplung an einen Hostrechner (hier K 1840) stehen ein serielles Interface (IFSS/V.24) und ein DMA-fähiges Parallelinterface zur Verfügung. Die Rate für die serielle Datenübertragung kann zwischen 50 und 19200 Baud gewählt werden. Das Terminal besitzt für die Bildschirmdarstellung vier Farbsysteme, die bis zu 16 Mio Farben darstellen können, sowie die Möglichkeit der Umwandlung der Farbwerte in 256 Graustufen.
Auf dem K 8919.11 wurden mehrere Programmsysteme vorgeführt, unter anderem das System UMSCHA und der Grafikeditor TEXgraf.
Das Programmsystem **UMSCHA** wurde von der Technischen Universität Dresden entwickelt, um Spannungen und Verzerrungen ermitteln zu können, wie sie beim Umformen doppelt gekrümmter Flächen beispielsweise im Automobilbau, im Schiffbau sowie in der Schuh- und Bekleidungsindustrie auftreten. Es dient auch als Hilfsmittel in der Geologie (Bild 3). Der Kraftaufwand und der Arbeitsbedarf der Umformvorgänge beliebigen Grades soll sowohl für das elastische als auch für das starrplastische und das elastisch-plastische Materialverhalten ermittelt werden können. In dialogorientierter Berechnung werden in kurzer Zeit Umformvorgänge einschließlich der Berücksichtigung der Wanddickenänderung und der elastischen Rückverformung ermittelt, geometrisch aufbereitet und die Ergebnisse grafisch dargestellt. Die Hardwarevoraussetzungen für das in Fortran programmierte System UMSCHA sind K 1840/IGT-2 (K8919) und das Betriebssystem SVP 1800.
Der grafische Schaltungseditor **TEXgraf** vom VEB Textimaelektronik ist als Bestandteil von Gate-Array-Entwurfssystemen ein wichtiges Hilfsmittel für den Entwicklungsingenieur. Gerade im Zeitalter der anwendungsspezifischen Schaltkreise kommt insbesondere komfortablen CAD-Systemen eine immer größere Bedeutung zu. TEXgraf wird dem durch eine einfache Bedienung und damit schnelle Erlernbarkeit gerecht. Eine hierarchische Menüführung mit vielen Automatikfunktionen in Fenstertechnik (siehe Bild 4) sowie wechselseitige Konvertierung zur alphanumerischen Beschreibungssprache NBS84 bieten gute Voraussetzungen, Schaltpläne zu erfassen, zu ändern und sowohl auf den Bildschirm als auch auf Plotter auszugeben. Die Schaltungseditierung kann dabei wahlweise grafisch oder alphanumerisch erfolgen, Plazieren und Trassierung können durch den Ingenieur selbst oder durch den Editor automatisch vorgenommen werden.
Eine Auflösung von 1280 × 1024 Bildpunkten gewährleistet eine gute Erkennbarkeit der Schaltungselemente und Symbole. Neben der enthaltenen, umfangreichen Symbolbibliothek bietet TEXgraf die Möglichkeit des Entwurfs eigener Symbole. Schaltungsprüfung und Darstellen der Simulationsergebnisse gehören ebenso zu den angebotenen Funktionen wie das Erzeugen und Modifizieren von Texten, grafischen Elementen und Farben. Im Rahmen von Gate-Array-Entwurfssystemen wie ARCHIMEDES werden die Vorzüge dieses grafischen Editors sicher schnell deutlich. Auf einem K 1840 bzw. einem dazu kompatiblen Rechner kann TEXgraf in Verbindung mit dem Terminal K 8919.11 genutzt werden.

Mit dem Übergang zur Serienproduktion des EC 1834 rückt das computerunterstützte Entwickeln immer mehr in die Nähe des Konstruktionsarbeitsplatzes. Der VEB Kombinat Robotron bietet mit **MultiCAD** ein für das Betriebssystem DCP entwickeltes, leistungsfähiges Paket an.
Aus den grafischen Elementen Punkt, Linie, Kreis, Kreisbogen, Band, Fläche, Text und Symbol werden die Zeichnungen zusammengestellt und können unter anderem in der Lage, der Form und der Anzahl verändert werden. Sowohl über das Grafiktablett als auch von der Tastatur können menügesteuert die Zeichnungen im Dialog eingegeben und editiert werden. Für die dreidimensionale Darstellung enthält MultiCAD Raumlinien und Raumflächen, die entstandenen Drahtmodelle werden in Parallelperspektive dargestellt und können aus verschiedenen Richtungen betrachtet werden.
In MultiCAD wurde eine Lisp-Version integriert, die einerseits zwar nur einen Teil des Sprachumfanges von Lisp enthält, andererseits aber um Grafikfunktionen erweitert wurde. Lisp gestattet es, nutzerdefinierte Kommandos zu erzeugen und wie systemeigene anzuwenden. Die Zeichnungen können zur Weiterverarbeitung in anderen Systemen in das Datenformat IGES bzw. in eine Textdatei konvertiert werden.
Als eine optimale Gerätekonfiguration für MultiCAD könnte beispielsweise gelten: EC 1834 (640 KByte RAM, 2 Floppy-Disk-Laufwerke, 1 Festplatte sowie 1 Arithmetikprozessor), alphanumerischer Monitor, grafischer Monitor K 7234 (color) oder K 7229.25 (monochrom), grafisches Tablett K 6405 und Plotter K 6416.

Ein überschaubares Menü sowie ein Tablett mit Stift und Lupe sollen die Arbeit mit **PCCAD**, einem leistungsfähigen, zweidimensionalen CAD-System des VEB Leitzentrum für Anwendungsforschung (LfA) Berlin für 16-Bit-PCs, nutzerfreundlich unterstützen.
Nach kurzer Einarbeitung beherrschbare, unterstützende mathematisch-geometrische Funktionen sollen ein professionelles Arbeiten gewährleisten. Die geometrischen Grundsymbole, wie beispielsweise Punkt, Strecke, Kreisbogen und Textzeile, können durch weitere anwenderspezifische Symbole ergänzt werden und können gespeichert werden; das Verbinden von Punkten, das Zeichnen von Tangenten, Winkelhalbierenden gehören zu den konstruktiven Grundfunktionen und werden exakt ausgeführt und gezeichnet.
Branchenorientierte Module für den Maschinenbau, die Elektrotechnik und Elektronik, die Architektur und die Geodäsie werden zum Grundmodul angeboten und bieten zusammen mit dem 16-Bit-PC EC 1834 und einem Colorgrafikbildschirm, einem Digitalisiertablett sowie einem Plotter und dem Betriebssystem DCP 3.2 effektive Konstruktionsmöglichkeiten.
Die Strukturierung einer Zeichnung ist in maximal 128 Ebenen für die Gliederung, die Bemaßung, das Schraffieren oder Ausfüllen einer Fläche möglich.

Das Forschungszentrum des Werkzeugmaschinenbaus und die Akademie der Wissenschaften der DDR stellten mit **CAD/CAM-NILES** ein durchgängiges rechnergestütztes System vor, das neben der Konstruktion auch die Erarbeitung der Technologie bis hin zur Programmierung numerisch gesteuerter Werkzeugmaschinen für die Produktion gebogener Teile aus Stahlblech ermöglicht. Eine höhere Flexibilität und eine deutliche Reduzierung des Teiledurchlaufs bei einer gleichzeitigen Steigerung der Produktivität der NC-Programmierung und der Produktion sind die wesentlichen Leistungsmerkmale. Die einzelnen Blechteile können als dreidimensionale Modelle erfaßt, dargestellt und zusammengestellt werden. Die visuelle Kontrolle unter den Einbaubedingungen (als 3D-Modell) und die Ableitung von Teilansichten aus dem 3D-Modell erleichtern die Arbeit.
CAD/CAM-NILES ermöglicht die Kontrolle, das Umwandeln und die Übernahme der Geometrie in das Programmsystem zur technologischen Vorbereitung. Die Geometrie aus dem CAD-System bildet die Grundlage für die NC-Programmierung der Einzelteile bis hin zu automatisierten Abläufen der Werkzeugauswahl und der Optimierung der Verarbeitungstechnologie.

Für den Leiterplattenentwurf mit dem EC 1834 zeigte das ZFTN des Kombinates Nachrichtenelektronik (Vertrieb: LfA) das System **PCLES** (Bild 5). Es ermöglicht das grafische Erfassen des Stromlaufplanes mit logischen oder Bauteil-Symbolen und das Erzeugen von Symbol- und Bauteillisten, von Transferdateien für den Übergang zum Layoutentwurf sowie von Testplots des Stromlaufplanes. Weiterhin sind die grafische Konstruktion des Leiterplattenumrisses, die Bauelementeplazierung, manuelles und automatisches Trassieren in bis zu 16 Lagen (maximal 1900 Trassen) und die Ausgabe von Steuerdaten für das Lichtzeichnen sowie für Bohrmaschine und Plotter möglich. Die maximale Leiterplattengröße beträgt 65 × 65 cm^2 (bei Raster 1/40"). Die maximal 511 Bauelemente können bis zu je 256 Pins besitzen. Durch die Kopplung mit einem K 1840 kann aus dem einheitlichen Bauelementekatalog des Kombinates die jeweils benötigte Teilmenge auf die Festplatte des PCs geladen werden.

Eine Grafik-Workstation für CAE- und CAD-Anwendungen auf der Basis des XT-kompatiblen EC 1841 bot die Akademie der Wissenschaften der Belorussischen SSR an. Herzstück ist der Grafikprozessor **MICROGRAPH GP/10** oder **GP/20**, an den der PC, ein Digitalisiertablett, ein Plotter, ein Tintenstrahldrucker und ein 19-Zoll-Grafikmonitor angeschlossen sind. Der GP/10 ermöglicht eine darstellbare Auflösung von 768 × 512 Punkten (adressierbar sind 1024 × 512 Punkte), 256 Farben aus einer Palette von 4096. Er arbeitet mit einem 8086-Prozessor und einem 8087-Koprozessor sowie 512 KByte RAM. Als Schnittstelle stehen ein schnelles Parallelinterface zu IBM-PCs, drei RS 232C-Schnittstellen und ein Centronics-Interface zur Verfügung. Der GP/20 besitzt zusätzlich einen Grafikcontroller in Bit-slice-Technik.

Die Farbgrafik-Workstation **Starstation** für CAD-, CAM- und CAE-Anwendungen (Bild 6) war im Angebot der Datagraph GmbH (BRD). Die Starstation besitzt eine 80286-CPU (12,5 MHz) mit Koprozessor 80287 (10 MHz) und 1 MByte RAM, ein Winchesterlaufwerk mit 42 MByte sowie eine 5,25-Zoll-Floppy mit 1,2 MByte. Die 1024 × 768 Bildpunkte können in 16 Farben (aus einer Palette von 4096) dargestellt werden. Der Bildspeicher umfaßt 4 Bildebenen und 640 KByte RAM. Optional können der Hauptspeicher auf 16 MByte erweitert, eine CPU 80386 (16 MHz) eingesetzt und eine 68-MByte-Winchesterdisk sowie ein Streamer angeschlossen werden. Der Bildspeicher ist auf 8 Ebenen und 1280 KByte erweiterbar; die Farbpalette kann auf 16,7 Mio Farben erhöht werden.
Mit der Starstation können Softwarepakete für den Anlagenbau, den Modell- und Formenbau, den Rohrleitungsbau, die mechanische Konstruktion und Fertigung, für die Elektrokonstruktion und die Konstruktion und Fertigung in der Elektronik und für Kinematik-Analyse sowie ein Datenbank- und Kopplungssystem geliefert werden. Außerdem wurde mit dem datagraph **Starset** (20-Zoll-Monitor Colorstar + Interfaceleiterkarte Pixelstar) die Möglichkeit angeboten, einen PC/XT, /AT oder PS/2-Modelle mit Mikrokanal zu einer Grafikworkstation mit den Leistungsmerkmalen der Starstation umzurüsten.

Gekoppelt mit dem **P8000 compact** vom Kombinat Elektro-Apparate-Werke Berlin stellte die Tektronix GmbH Köln das Netzwerk-Farbgrafikterminal **Tek 4211** aus (Bild 7). Es ist mit zwei 32-Bit-Prozessoren bestückt: Den schnellen Bildaufbau und die hohe Auflösung (1024 × 768 Punkte) verdankt es einem neuen Grafikcontroller mit dem 34010 von Texas Instruments und drei Gate-Arrays von Tek. Ein- und Ausgabe sowie das Datenmanagement werden vom i80386 SX gesteuert.
Auf dem 15-Zoll-Bildschirm können 16 Farben aus einer Palette von 4096 dargestellt werden. Außerdem sind drei RS 232C-Anschlüsse, eine Centronics-Schnittstelle, 0,75 MByte RAM und eine VT200-Tastatur vorhanden. Neben einem breiten Softwareangebot ist optional ein 19-Zoll-Bildschirm, 256 aus 16,7 Mio Farben,

ein Ethernet-Anschluß, ein Koax-Anschluß mit IBM-kompatibler Tatstatur, zusätzlich 2 MByte RAM, eine Maus sowie Rändelräder möglich.

Die Siemens AG hatte schon vor Jahren zur durchgängigen Informationsverarbeitung im gesamten Unternehmen ihr umfassendes Konzept CAI – Computer Aided Industrie entwickelt. Ein Bestandteil dessen ist das in Leipzig gezeigte Softwaresystem **SILINE** auf der Bürocomputerbasis C 30 unter dem Betriebssystem BS 2000. Auf einem Grundsystem können die jeweils benötigten Softwarekomponenten wie Bausteine so zusammengefügt werden, daß bei Erweiterungen die bereits vorhandenen Daten erhalten bleiben; eine durchgängig einheitliche Benutzeroberfläche erleichtert dabei die Systembedienung. SILINE 200 bietet zur Zeit die betriebswirtschaftlichen Bausteine PP für die flexible Produktion, PU für wirtschaftliche Beschaffung, SA für den Vertrieb, PM für das Personalwesen und AC für eine entscheidungsorientierte Buchhaltung. Zur informationstechnischen Verbindung von Fabrik, Büro und Umwelt gibt es darüber hinaus Kopplungsschnittstellen zum Werkstattsteuerungssystem BORA-X, zum Bürokommunikationssystem OCIS und zum CAD-System CADIS. Mittels **CADIS** wurde in Leipzig auf der Grundlage des grafischen Arbeitsplatzes 9733 und der BS-2000-Rechner C 30 der Ablauf einer mechanischen Konstruktion gezeigt (Bild 8). Für unterschiedliche Bearbeitungen stehen dabei Softwareprodukte zum Beispiel für Drehteile, Bohr- und Frästeile oder Blechteile als Verfahrenskette zur Verfügung; auch gibt es Software zur komfortablen Variantenkonstruktion. Selbstverständlich stehen am Ende der Kette NC-Programme zur Steuerung der Maschinen.

Computertechnik

Auch in diesem Jahr hatte eine Vielzahl von Ausstellern neue Hardware zu bieten. Zwar war die Fülle von PCs verschiedenster Länder des vergangenen Jahres diesmal nicht zu beobachten – die beeindruckende Präsentation Taiwans fehlte beispielsweise völlig –, doch waren erstmals westliche Firmen mit 32-Bit-Computern vertreten. So wurde der Generalsekretär des ZK der SED und Vorsitzende des Staatsrates der DDR, Erich Honecker, während seines Messerundganges vom Botschafter der USA, Richard Clark Barkley begrüßt, der seinen Gast über die von renommierten US-Firmen präsentierten Hochtechnologien informierte. Als Beispiel nannte der Botschafter den 32-Bit-Computer der Firma Intraco, der zur Frühjahrsmesse erstmals in Osteuropa zu sehen war. (Dieser Computertyp ist als Zentraleinheit übrigens Bestandteil des im folgenden genannten Rechnersystems Adria.)

Die jugoslawische Firma Iskra Delta stellte in Leipzig einen neuen DEC-kompatiblen 32-Bit-Rechner vor. Der **ADRIA DOA-V31** (Bild 9) besitzt eine CPU MicroVAX II von der Digital Equipment Corporation (DEC), einen Gleitkomma-Koprozessor, 1 MByte RAM, eine 335-MByte-Harddisk, einen Streamer und eine Echtzeituhr. Er wurde für eine große Zahl von Anwendungen, wie Echtzeitanwendungen, Büroautomatisierung und Programmentwicklung ausgelegt. Er arbeitet mit einem erweiterten Q-Bus und benutzt als Betriebssystem das UNIX-kompatible DELTA/V. Der ADRIA kann als Teil eines Netzwerkes durch Verwendung eines Ethernet-Q-Bus-Interfaces betrieben werden.

Auch die Firma IBM war mit einem vielseitig einsetzbaren Minicomputersystem in Leipzig vertreten. Das Mehrplatzsystem **IBM 5363** eignet sich vorrangig für dezentrale Rationalisierungsaufgaben, bietet aber auch die Möglichkeit der Kopplung mit Großrechnern über eine Datenfernübertragung oder mit anderen Rechnern über LAN-Anschlüsse. Demonstriert wurde der Einsatz des IBM 5363 als Fileserver (Bild 10). Er kann mit 16 lokalen Datenstationen (28 mit Erweiterung) sowie mit bis zu 64 entfernten Datenstationen arbeiten. Er besitzt zwei bis vier Prozessoren, 1 oder 2 MByte RAM, eine Festplattenkapazität von 65 bis 420 MByte und ein 5,25-Zoll-Diskettenlaufwerk mit 1,2 MByte. Durch seine Kompatibilität mit der IBM System/36-Familie steht für den IBM 5363 eine große Softwareauswahl zur Verfügung.

Elorg-Data Helsinki stellte die PDP-11-kompatiblen 16-Bit-Minicomputer im Towergehäuse E-60/ED und E-102 (siehe Bild unten) aus. Sie stellen eine sowjetisch-finnische Gemeinschaftsarbeit dar. Der **E-60/ED** enthält einen RAM von 256 KByte bis 4 MByte, ein 5,25-Zoll-Floppylaufwerk mit 1 MByte, eine 5,25-Zoll-Harddisk mit 20 bis 80 MByte und einen Streamer mit 20 bis 60 MByte. Als serielle Schnittstelle dienen RS 232 und 20-mA-Stromschleife. Der E-60/ED ist kompatibel zu den Minirechnern CM4, CM5 und CM1420. Seine Betriebssysteme sind OS-RW, MIOS, RAFOS/FODOS und das UNIX-kompatible INMOS.
Der **E-102** arbeitet mit dem Mikroprozessor J11, einem RAM von 512 KByte bis 4 MByte, einer 5,25-Zoll-Floppy-Disk mit 800 KByte, einer 5,25-Zoll-Harddisk mit 40 bis 300 MByte und mit einem 40-MByte-Streamer. Er ist mit dem E-60/ED kompatibel; seine Betriebssysteme sind OS-RVM, MIKROS, FODOS 3 und UNIX/VENIX.

Der VEB Elektro-Apparate-Werke „Friedrich Ebert" Berlin-Treptow war der erste Betrieb der DDR, der mit





Fotos: Hemke, Weiß

dem **P8000 compact** (siehe Bild oben) einen Mikrorechner mit dem neuen 16-Bit-Prozessorsystem U80600 entwickelt hat. Der P8000 compact und das Prozessorsytem U 80600 wurden bereits in MP 5/1989 ausführlich vorgestellt.

Der niederländische Computerproduzent Tulip, mit dem beziehungsreichen Namen und Firmenlogo, stellte erstmals in Leipzig aus und hatte mehrere seiner Modelle mitgebracht. Die Firma wurde 1976 als Beratungsunternehmen für den professionellen Mikrocomputereinsatz gegründet und produziert heute ein breites Spektrum, vor allem im PC-Sektor. Nach Steigerungsraten von jährlich etwa 200 % sollen in diesem Jahr 300 000 PCs produziert werden; 1987 waren es 31 000. Dabei setzt die Firma auf fortschrittliche Technik und Entwicklung, beispielsweise mit dem Einsatz selbstentwickelter ASICs. Nach eigenen Angaben war Tulip vor etwa zwei Jahren der erste europäische Anbieter eines PCs mit dem 80386-Prozessor und vor etwa einem Jahr eines PCs mit dem 80386SX-Prozessor. Über den at386sx (im Bild 11 rechts) werden wir zu einem späteren Zeitpunkt etwas ausführlicher berichten; zunächst zu den ebenfalls gezeigten Modellen pc compact 2 und at compact 2. Die Namen deuten bereits auf das Mini-Maß der Gehäuse hin: 30,5 cm breit, 37,5 cm tief und 14 cm hoch. Trotzdem ist noch Platz für 5 Steckplätze voller Länge (beim at compact 2 ein 8-Bit-PC-Steckplatz und vier 16-Bit-AT-Steckplätze). Der **pc compact 2** arbeitet mit dem 8088-kompatiblen Prozessor V20 von NEC und ist dank maximaler 9,54 MHz (umschaltbar auf 4,77 und 7,15 MHz) und Zero Wait States schneller als viele andere Kompatible. Als Extras wurden in Leipzig die Grafikmöglichkeiten im VGA-Standard demonstriert.
Auch der **at compact 2** (im Bild 11 links) zählt mit 12,5 MHZ bereits zu den schnellen 286er ATs, ebenfalls ohne Wartezyklen arbeitend. Der RAM beträgt 640 KByte. Diskettenlaufwerke können im 5,25-Zoll-Format (360 KByte oder 1,2 MByte) und 3,5-Zoll-Format (720 KByte oder 1,44 MByte) verwendet werden; 3,5-Zoll-Harddisks gibt es mit 20 MByte (30 ms Zugriff, ST 506), 40 MByte (25 ms Zugriff, ST 506), und 60 MByte (25 ms Zugriff, RLL). Neben dem Mausanschluß gibt es eine serielle Schnittstelle (RS 232C), ein paralleles Centronics-Interface sowie einen weiteren parallelen Anschluß des Dual-Grafikadapters für einen zusätzlichen Monitor. Dies kann ein Monochrom- oder ein RGB-Monitor sein, wobei Hercules und CGA unterstützt werden. Vorhanden ist auch ein Enhanced Grafikadapter (TEVA) für das EGA-Format mit 640 × 350 Punkten Auflösung. Zum Standardlieferumfang der Software gehören das Betriebssystem MS-DOS 3.x, der GW-Basic-Interpreter mit grafischen Möglichkeiten, MS-Windows als benutzerfreundliche Bedienoberfläche, das Textverarbeitungsprogramm MS-WRITE sowie der Grafikeditor MS-PAINT.

Im vergangenen Jahr war die Firma Ensch Computer Service mit einem XT-kompatiblen PC vertreten; zur diesjährigen Messe hatte das Handelshaus aus Luxemburg einen 286er Laptop und als Spitzenprodukt einen AT-kompatiblen Tower mit einem 386er Prozessor mitgebracht. Für den Entec **EPC 386** (Bild 12, links) bietet sich aufgrund der hohen Leistung der Einsatz als Fileserver oder als Stand-alone-Workstation an. Der Prozessor 80386 läuft mit 16 bzw. 20 MHz und ist wahlweise hardware- und softwareumschaltbar. Ein Steckplatz für den 80287/80387-Koprozessor ist vorhanden. Der 2-MByte-RAM ist auf 4 oder 8 MByte erweiterbar; der ROM beträgt 64 KByte. Als Massenspeicher können darüber hinaus 5,25-Zoll-Floppies mit 1,2 MByte, 3,5-Zoll-Floppies mit 1,44 MByte und eine Festplatte mit bis zu 80 MByte genutzt werden. Die Software enthält MS-DOS 3.3 und GW-Basic.
Der Laptop Entec **EPC LA 20** (Bild 12, rechts) ist mit einem 80286-Prozessor mit 10 MHz ausgestattet und hat einen Sockel für den 80287-Koprozessor. Das 5,9 kg schwere Gerät beinhaltet bereits zwei von rechts zugängliche 3,5-Zoll-Floppylaufwerke mit je 720 KByte sowie als Option eine 20-MByte-Festplatte. Der RAM umfaßt 640 KByte, erweiterbar um 2 MByte, und der ROM 32 KByte. Das Supertwist-LCD ermöglicht eine Auflösung von 640 × 200 Punkten; ein Anschluß für einen externen Farbmonitor (640 × 200 Punkte) oder einen Monochrommonitor (720 × 348 Punkte) ist vorhanden.

Der japanische Hersteller Epson war auch in diesem Jahr wiederum nicht nur mit Druckern vertreten – die seinen weltweiten Ruf begründeten –, sondern auch mit einer reichhaltigen Auswahl von Computern. Beispiels-

weise mit dem schon im Vorjahr gezeigten und jetzt mit Messegold ausgezeichneten modularen Handheldcomputer PX-16 (vorgestellt in MP 9/88) und dem ersten PS/2-kompatiblen PC von Epson, dem **PSE 30** (Bild 13). Letzterer ist ein zum PS/2 Modell 30 voll kompatibler PC – also mit dem Prozessor 8086-1 (8 oder 10 MHz; ohne Wartezyklus) und ohne Mikrokanal. Die gelungene Gestaltung des kompatiblen PCs wurde mit der Medaille für gutes Design honoriert. In dem nur etwa 35 × 40 × 12 cm^3 (B × T × H) großen Gehäuse haben parallel zur Mutterplatine (ca. 19 × 28 cm^2) noch vier lange Steckkarten Platz. Als Floppies werden 1 oder 2 3,5-Zoll-Laufwerke mit 720 KByte, als Harddisk ein 20-MByte-Laufwerk verwendet. Hier wird die Hürde deutlich, die aus den bisher üblichen 5,25-Zoll-Disketten resultiert und durch vorheriges Überspielen auf 3,5-Zoll-Disketten gemeistert werden muß. Der RAM im PSE 30 umfaßt 640 KByte, der ROM 64 KByte. Die Bildschirmausstattung enthält als Coloradapter standardmäßig MCGA (Multi Color Graphics Array) und als Option VGA (Video Graphics Array).
Weiterhin zeigte Epson seinen XT-kompatiblen **PC-Portable** (Bild 14), dessen V30-Mikroprozessor von NEC kompatibel zum 8086 ist und von 4.77 auf 10 MHz umgestellt werden kann. Als Standard enthält der Portable ein 3,5-Zoll-Diskettenlaufwerk mit 720 KByte; als Option ein zweites oder eine 20-MByte-Festplatte. Um auch 5,25-Zoll-Disketten nutzen zu können, läßt sich die Centronics-Schnittstelle als Anschluß für ein externes Laufwerk umprogrammieren. Im Gerät sind zwei 8-Bit-Steckplätze im Epson-Format vorhanden, wobei ein Platz mit dem Festplattencontroller zu belegen ist. Wenn über den RGB-Anschluß ein externer Farbmonitor verwendet werden soll, läßt sich das hintergrundbeleuchtete Supertwist-LCD (640 × 200 Punkte, CGA) abnehmen.
Der RAM des Portable hat 640 KByte, der ROM 16 KByte. Der Betrieb erfolgt wahlweise am Netz, über einen 12-V-Adapter oder mit den integrierten acht NiCd-Akkus, die bei Floppybetrieb etwa 5 Stunden und bei Festplattenbetrieb etwa 2 Stunden lang den notwendigen Strom liefern (Ladezeit 12 Stunden).

Zum zweitenmal auf der Leipziger Messe vertreten war das brasilianische Unternehmen Itautec, diesmal mit Druckern und einem XT-kompatiblen PC. Der **IS 30 plus** (Bild 15) enthält einen 8086-2-Mikroprozessor, der mit 4,77 oder 8 MHz getaktet werden kann (ohne Wartezyklus). Nachrüstbar ist der Arithmetikprozessor 8087-2. In der Standardausstattung enthält der RAM bereits 896 KByte, wobei 256 KByte für den Bildschirm genutzt werden. Bei Verwenden des EMS-Standards kann der RAM auf Erweiterungskarten bis zu 6 MByte ausgebaut werden. Insgesamt sind 3 Standard-PC-Slots verfügbar sowie eine parallele (Centronics) und zwei serielle (RS 232C) Schnittstellen. Standardmäßig ist ein 12-Zoll-Schwarzweißmonitor im EGA-Modus (640 × 350 Punkte) mit 16 Graustufen; möglich sind aber auch CGA, MDA, VGA, Hercules und Plantron. Als Betriebssystem wird das MS-DOS-kompatible SISNE Plus 3.3 mitgeliefert.

Mit den auf der 4. Umschlagseite vorgestellten PCs wird ein Wandel deutlich, der sich bei vielen Herstellern abzeichnet. Sowohl Schneider als auch Commodore und Atari sind seit geraumer Zeit bestrebt, sich vom Image eines Heimcomputerproduzenten zu lösen und im Profi-PC-Markt Fuß zu fassen. Bei der Schneider AG, die erstmals in Leipzig ausstellte, wurde diese Entwicklung 1986 eingeleitet mit den PC-kompatiblen Modellen PC 1512 und PC 1640. Ein neues Konzept hatte im Vorjahr dann den EuroPC und den Tower AT zur Folge; inzwischen ist auch ein Laptop im Programm, allerdings nicht aus eigener Entwicklung und Produktion.
Der **EuroPC** (Bild 16) wird als „Einsteiger-PC" angesehen und fällt durch seine eigenwillige Gestaltung auf. Das Grundgerät enthält die Tastatur und ein 3,5-Zoll-Diskettenlaufwerk mit 720 KByte. Außerdem die CPU 8088-1 mit 9,54 / 7,16 / 4,77 MHz, 512 KByte RAM und 32 KByte ROM. Andere Massenspeicher, die auch von Schneider beziehbar sind, müssen extern betrieben werden. Beispielsweise ein weiteres 3,5-Zoll- oder ein 5,25-Zoll-Floppylaufwerk und eine externe 20-MByte-Harddisk, wofür je ein Anschluß vorhanden ist. Außerdem gibt es je eine parallele (Drucker), serielle (RS 232C) und Maus-Schnittstelle. Platz ist im Grundgerät außerdem noch für eine 8-Bit-Steckkarte halber Länge. Bei Verwendung eines 12-Zoll-Schwarzweißmonitors wird die Herculesgrafik genutzt (720 × 348 Punkte), bei einem 14-Zoll-Colormonitor die CGA-Grafik mit 640 × 200 Punkten bei 4 Farben, 320 × 200 Punkten bei 16 Farben und im Alphamodus mit 40 × 25 oder 80 × 25 Zeichen und 16 Farben. Die Software umfaßt MS-DOS 3.3, MS-GW-Basic und MS-Works für Textverarbeitung, Kalkulation, Datenbankbetrieb, Grafik und Kommunikation.

Zur Abrundung der Produktpalette dient seit kurzem der **Portable AT** (Bild 17). Charakteristisch ist bei dem Modell der Plasmabildschirm mit einer maximalen Auflösung von 640 × 400 Punkten (CGA mit Doublescan); außerdem läßt sich ein externer Farbmonitor anschließen. Die 80286-CPU arbeitet mit 8 MHz, optional ist ein 80287-Koprozessor. Der 640-KByte-Standard-RAM ist mittels RAMcard auf 2,6 MByte erweiterbar; der ROM beträgt 32 KByte. Ein 3,5-Zoll-Diskettenlaufwerk (720 KByte) und eine 20-MByte-Festplatte sind bereits eingebaut, extern läßt sich ein weiteres 3,5-Zoll- oder ein 5,25-Zoll-Diskettenlaufwerk (1,2 MByte) betreiben. Außerdem gibt es eine parallele und eine serielle Schnittstelle. Für Erweiterungen ist noch ein freier Slot vorhanden. MS-DOS 3.3, MS-GW-Basic und MS-Works gehören zum Lieferumfang.

Bis zur diesjährigen CeBIT in Hannover – wo erstmals ein 386er PC gezeigt wurde – stellte der **Tower AT** (Bild 18) das Spitzenprodukt in Schneiders neuer PC-Linie dar. Auch dieses Gerät hebt sich dank seiner äußeren Erscheinung von anderen Kompatiblen deutlich ab. So werden die Erweiterungsmodule seitlich an das Grundgerät „angedockt". Während in den Versionen 201, 202 und 220 die 80286-CPU mit 10 MHz verwendet wird, ist das Modell Tower AT 260 mit einer 12,5-MHz-CPU ausgestattet; jeweils ohne Wartezyklen arbeitend. Der Hauptspeicher umfaßt 512 KByte DRAM, auf der Hauptplatine auf 1 MByte erweiterbar, sowie 96 KByte ROM mit dem Phoenix- und EGA-BIOS. Weitere Speicher sind ein 3,5-Zoll-Diskettenlaufwerk mit 720 KByte (1,44 MByte beim AT 260) und eine 3,5-Zoll-Festplatte mit 20 MByte (60 MByte, 25 ms Zugriff beim AT 260). Der modulare Aufbau erlaubt neben 4 freien Steckplätzen (3 × 16 Bit AT, 1 × 8 Bit PC, beim AT 260 1× durch Festplatten-Controller belegt) für Erweiterungskarten den externen Anschluß von einem 5,25-Zoll-Diskettenlaufwerk (1,2 MByte) und einem Streamer mit 40 MByte. Darüber hinaus gibt es eine parallele (Drucker), eine serielle (RS 232C) und eine Maus-Schnittstelle. Die Hauptplatine enthält einen Multifunktions-Grafikchip als Videoadapter für folgende Modi: MDA, Monochrom-Hercules, CGA, EGA und Hi-Res-Grafik. Bei der hochauflösenden HiRes-Grafik (High Resolution) bedeutet das beispielsweise unter Verwendung eines Multiscanmonitors eine Auflösung von bis zu 800 × 600 Punkten (16 aus 64 Farben). Die Bildwiederholfrequenz von 70 Hz gewährleistet dabei ein flimmerfreies Bild. Durch die automatische Erkennung des angeschlossenen Monitors (monochrom, CGA oder Multiscan) erübrigen sich die sonst üblichen DIP-Einstellungen. Als Bildschirm stehen 12-Zoll-Monochrom-, 14-Zoll-CGA- bzw. -EGA- und ein Hochleistungs-Multiscan-Monitor zur Auswahl. Zum Systemumfang gehören auch beim TowerAT die Software MS-DOS 3.3, MS GW-Basic und MS-Works.

Sowohl mit der Produktpalette von Commodore als auch mit der Organisationsstruktur des Unternehmens sieht sich Winfried Hoffmann, Vice President Central European Region, wie er auf einer Pressekonferenz erklärte, gut gerüstet für die 90er Jahre. Zwar hatte die Firma 1988 noch einmal einen nicht mehr erwartet hohen Absatz des Heimcomputerveteranen C 64 zu verzeichnen, doch belegt die Entwicklung des UNIX-Amiga, des Transputerboards für den Amiga und von 386er PCs den Anspruch Commodores im Profibereich. In Leipzig wurden auf einem vergrößerten Stand die Modelle C 64, C 128, PC 10-III/20-III, PC 40-III, Amiga 500 und Amiga 2000 gezeigt. Hier die Daten des neuen Commodore-ATs, des **PC 40-III** (Bild 19). Er stellt den überarbeiteten Nachfolger des PC 40 dar und fällt vor allem durch das verkleinerte Gehäuse auf (146 × 381 × 356 cm^3). Damit verbunden ist allerdings die Einschränkung auf 4 freie Steckplätze (1 × PC, 3 × AT). Die Taktfrequenz des 80286-Prozessors ist zwischen 6, 8 und 12 MHz umschaltbar (daher III), ein Sockel für den 80287 ist auf der Platine vorhanden. Standardmäßig ist 1 MByte RAM, so daß neben den 640 KByte MS-DOS 384 KByte beispielsweise als RAM-Disk genutzt werden können. Der RAM läßt sich auf der Hauptplatine bis 8 MByte erweitern. Ebenfalls auf der Hauptplatine befinden sich jetzt Video-Controller und Grafik-Adapter. Durch Einsatz des VGA-Chips PVGA von Paradise ist auf dem standardmäßigen 14-Zoll-Monochrommonitor eine Auflösung von 924 × 400 Punkten bei einer Bildwechselfrequenz von 70 Hz möglich. Die Modi MDA, CGA, Hercules und EGA werden emuliert (das Foto zeigt den 40-III mit Wyse-Monitor und VGA-Darstellung).
Als Massenspeicher sind eine 40-MByte-Festplatte (28 ms schnell) sowie ein 5,25-Zoll-Diskettenlaufwerk mit 1,2 MByte eingebaut; als Option gibt es auch ein 3,5-Zoll-Diskettenlaufwerk mit 720 KByte oder 1,44 MByte. Zu der seriellen und der parallelen Schnittstelle kam nunmehr ein Maus-Interface hinzu.

Mit einer Vielzahl neuer Computer ergänzte Atari die Produktpalette sowohl im ST-Bereich als auch bei kompatiblen PCs. Es werden jetzt zwei in sich geschlossene Linien geboten, die alle Nutzergruppen – vom Einsteiger bis zum Profianwender – berücksichtigen sollen. Die Eckpunkte dürften dabei der neue „erste MS-DOS-kompatible Pocketcomputer" PC Folio und die Transputer Workstation ATW bilden – beides übrigens Geräte, die zwar in den Presseinformationen angekündigt, jedoch nicht auf dem Ataristand präsent waren. Die wichtigste Produktgruppe bleiben für Atari in der BRD die ST-Modelle mit 71,6 Prozent des Umsatzes, gefolgt von MS-DOS-Computern mit 12,2 Prozent. Zwar konnte das Betriebsergebnis der Atari Computer GmbH 1988 um 8,7 Prozent verbessert werden, dennoch meinte Geschäftsführer Alwin Stumpf: „Die 1988 erzielten Zahlen stellen nicht das wahre Marktpotential dar, wenn man berücksichtigt, daß wir monatelang mit den Lieferungen hinter den Bestellungen zurücklagen, weil die Verfügbarkeit von DRAMs nicht mit der Nachfrage Schritt halten konnte".
Mit dem neuen **PC 5** (Bild 20) war Atari zur LFM einer der ersten Aussteller von 32-Bit-Personalcomputern. Der PC 5 verfügt über den Intel-80386-Prozessor, der bis zu 16 MHz getaktet werden kann. Der Hauptspeicher umfaßt 2 MByte RAM, der auf der Hauptplatine bis zu 8 MByte erweiterbar ist. Der Cache als schneller Zwischenspeicher hat 64 KByte. Die Massenspeicher sind eine 5,25-Zoll-Floppy mit 1,2 MByte und eine 60-MByte-Festplatte (28 ms Zugriffszeit). Als Erweiterungssteckplätze sind im Gehäuse 4 AT-Slots und 1 PC-Slot vorhanden. Verfügbare Videoadapter sind CGA, EGA, MDA, Hercules und VGA. Als Interfaces gibt es eine parallele Centronics- und zwei serielle RS-232C-Schnittstellen. Zum Lieferumfang gehören MS-DOS 3.3, MS-GW-Basic und MS-Windows 386.

H. Hemke, H.-J. Hill, H. Weiß



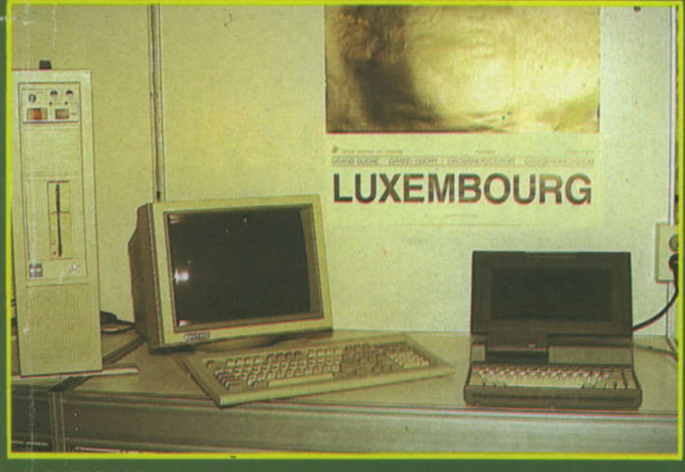
9 *32-Bit-Mikrocomputer ADRIA von Iskra Delta*



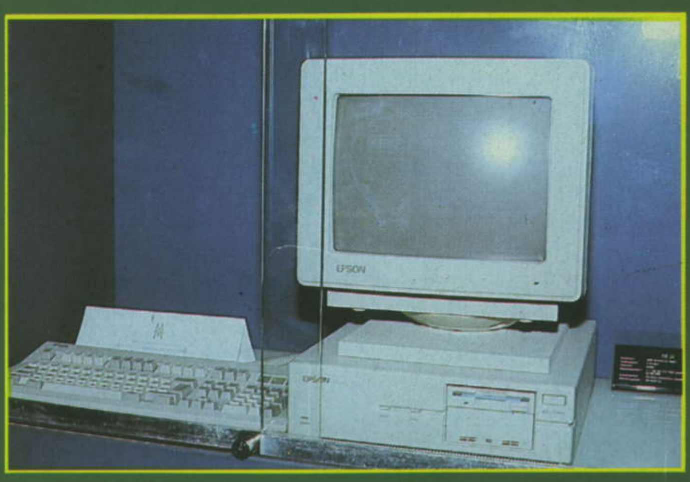
10 *Mehrplatz-Computersystem IBM 5363*



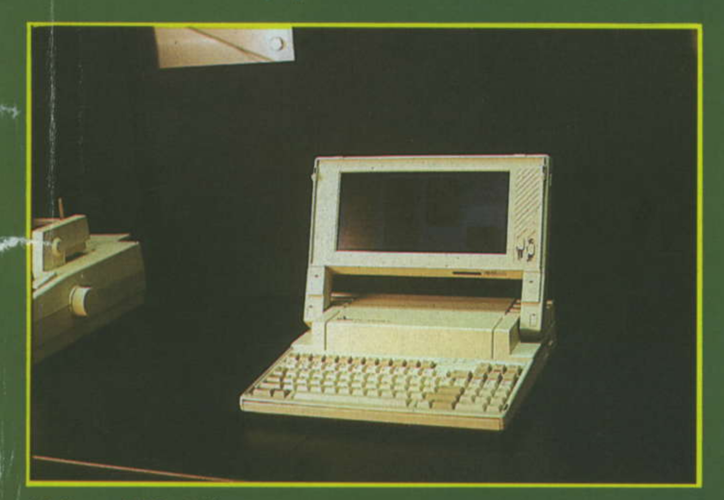
11 *Im Zeichen der Tulpe: Der Stand von Tulip*



12 *ENTEC EPC 386 und Laptop LA 20*



13 *Epson PSE 30*



14 *Epson PC Portable*



15 *Itautec IS 30 plus*

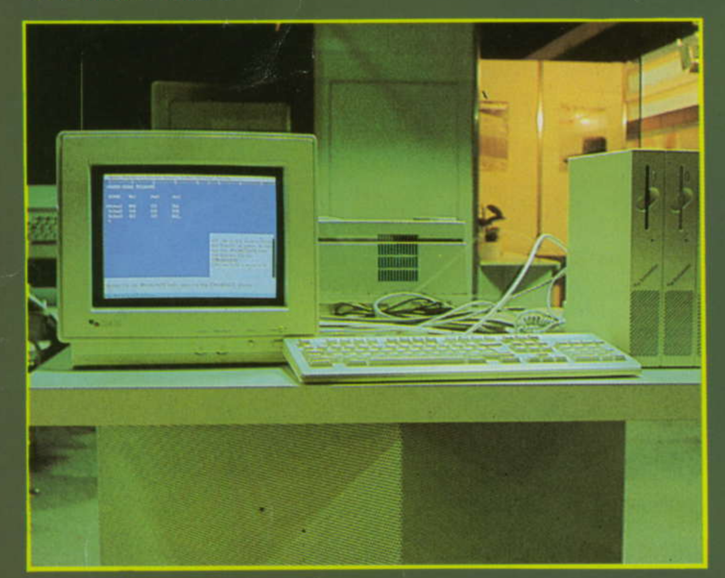
ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 7
S. 193–224



16 *Schneider EuroPC*

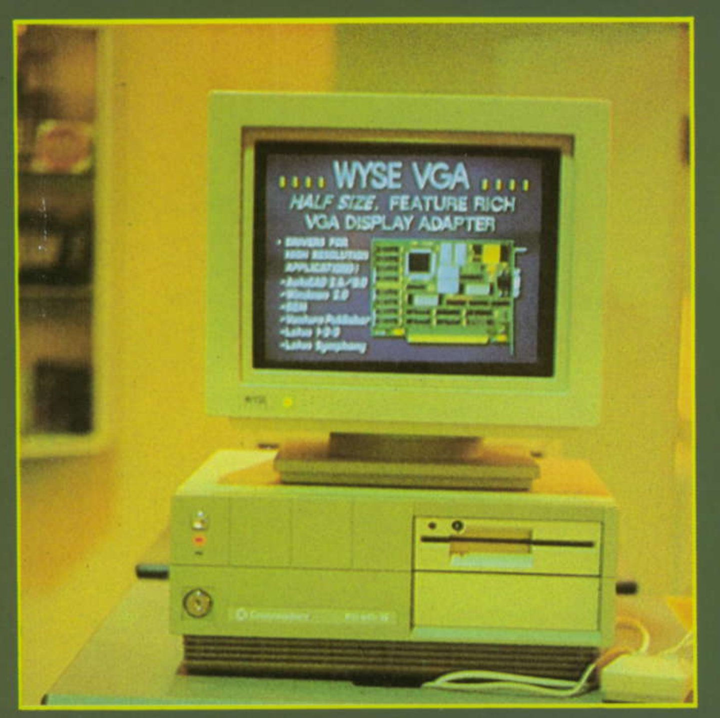
17 *Schneider PortableAT*

Lesen Sie hierzu unseren Bericht von der LFM '89



18 *Schneider TowerAT*

LFM '89



19 *Commodore PC 40-III*



20 *ATARI PC 5*

ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK

MP

Heft 8 · 1989

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0232-2892

Kopplung von UNIX-Rechnern

Portabel programmieren mit C

Diskrete Fouriertransformation

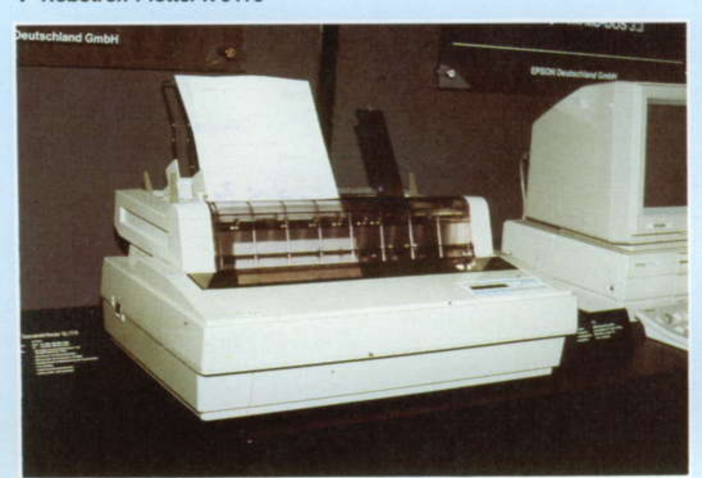
Maus und Turbo-Pascal



1 *Robotron-Plotter K 6416*



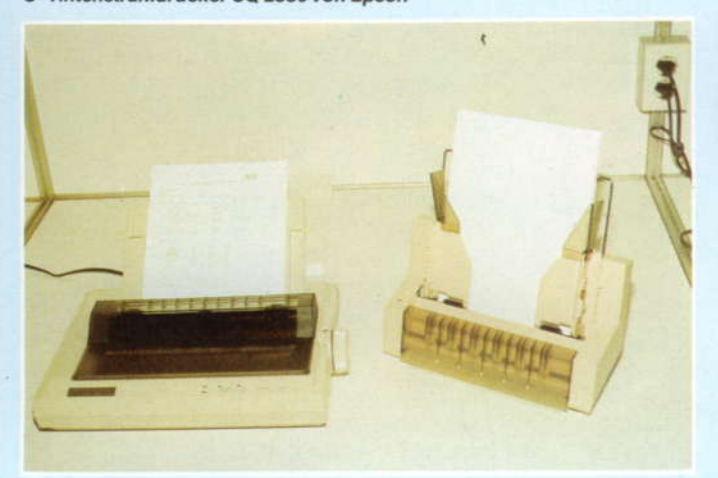
2 *Robotron-Schreibmaschine Erika S 6007*



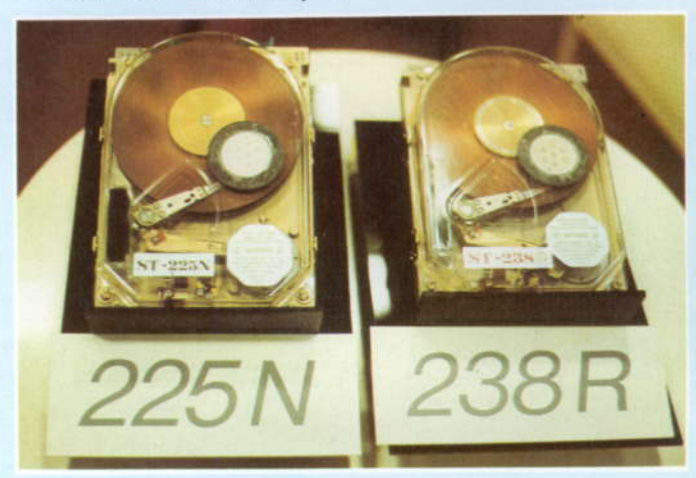
3 *Tintenstrahldrucker SQ 2550 von Epson*



4 *Laserdrucker GQ 5000 von Epson*



5 *Seikosha-Nadeldrucker SP 1600 AI*



6 *Festplattenlaufwerke von Seagate*



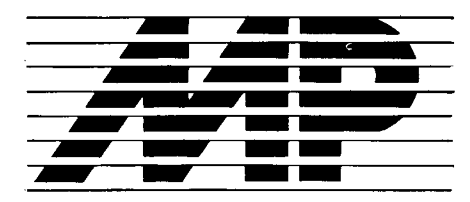
7 *Böwe-Computerarbeitsplatz mit Schneidautomat*



8 *5,25-Zoll-Floppybeistellgefäß von IBM*

Lesen Sie dazu unseren Bericht von der LFM '89

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 8 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titelfoto Hans Weiß

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß 14. Juni 1989

AN (EDV) 49837

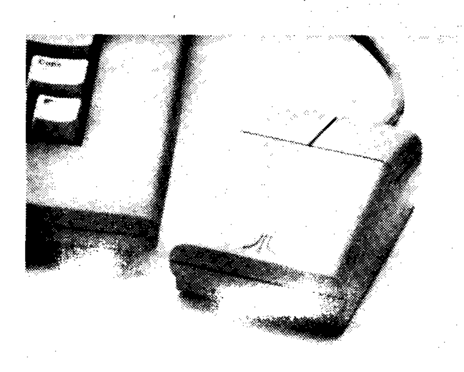
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig

Zum Titelblid

In der DDR stehen sowohl für verschiede 16- und 32-Bit-Rechner als auch für ESER-Rechner UNIX-kompatible Betriebssysteme zur Verfügung. Für die Kopplung dieser Rechner können Sie zwischen zwei Möglichkeiten wählen. Entweder Sie machen es so wie auf unserem Titelbild, oder Sie entschließen sich, unseren Beitrag „Kopplung von UNIX-Rechnern“ auf der **Seite 227** zu lesen. Zwar ist die erste Methode schneller, dennoch empfehlen wir die zweite.



Mit dem Beitrag „Maus-Anwendung in Turbo-Pascal“ auf der **Seite 232** stellen wir Ihnen Prozeduren vor, die in Turbo-Pascal-Anwenderprogramme eingebunden werden können. Damit wird dem Nutzer dieser Programme die komfortable Mausanwendung ermöglicht.



Nachdem wir im ersten Teil unseres Berichtes von der LFM '89 (in MP 7/1989) über CAD/CAM-Technik und -Anwendungen und über Computertechnik berichteten, befaßt sich der zweite und abschließende Teil auf der **Seite 253** mit peripherer Hardware, Bauelementen und Software.

Vorschau

Für das Heft 9/1989 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Datenanalyse mit dynamischer Grafik
- Heap-Nutzung unter Turbo-Pascal
- Kursormanipulation

Inhalt

Schlüsseltechnologien auf Moskauer Ausstellung

Eine internationale Ausstellung „Rechentechnik und Informatik" sozialistischer Länder war im Mai und Juni in Moskau zu sehen. Auf dem Gelände der Volkswirtschaftsausstellung zeigten Betriebe, Kombinate und Vereinigungen aus Bulgarien, der ČSSR, der DDR, Kuba, Polen, Rumänien, der UdSSR und Ungarn neueste Erzeugnisse auf dem Gebiet von Rechentechnik und Informatik. Der Tag der DDR wurde am 2. Juni begangen. Höhepunkt war die Unterzeichnung eines Vertrages zwischen den Außenhandelsbetrieben Robotron Export-Import und Elektronorgtechnika über Lieferungen von Robotron-Rechentechnik im kommenden Jahr für mehr als 100 Millionen Rubel.

Im Pavillon 2 auf dem Gelände der Volkswirtschaftsausstellung konnten sich die Besucher am DDR-Stand mit der Leistungskraft des VEB Kombinat Robotron vertraut machen. Besonderes Interesse fanden der 32-Bit-Rechner und das Demonstrationszentrum über den ersten gemeinsamen Software-Betrieb der DDR und der UdSSR „Zentron" in Kalinin. *ADN*

100. ICA-Rechner aus dem EAB

Pünktlich zur Kommunalwahl lösten die Werktätigen des VEB Elektroprojekt und Anlagenbau Berlin (EAB) eine für die Volkswirtschaft bedeutende Verpflichtung ein: Am 5. Mai wurde der 100. Industriecomputer fertiggestellt und in den 48-Stunden-Test im Prüffeld überführt.

Der 16-Bit-Rechner ICA 700 ist gegen rauhe Produktionsbedingungen wie Hitze, Staub und Vibration besonders geschützt. Die ersten Gerätesysteme sind in der Republik im Einsatz. Sie verhelfen zum Beispiel dem Werkzeugmaschinen-Kombinat „Fritz Heckert" Karl-Marx-Stadt, dem Schwermaschinenbaukombinat „Ernst Thälmann" Magdeburg, den Uhrenwerken Ruhla und dem VEB Schiffselektronik Rostock zur Automatisierung der Fertigung. Den Beschäftigten im EAB ist es zu danken, daß im Mai die Serienproduktion begann und bis Jahresende 300 weitere Computer ausgeliefert werden. 1990 soll sich die Produktion durch Ausbau und Verbesserung der Technologie verdoppeln. *ADN*

Fertigungskapazitäten für „dekotronik" werden ausgebaut

Das im Rationalisierungsmittelbau des Kombinats Deko entwickelte und bereits erprobte Erfassungssystem für Prozeßdaten „dekotronik" bewährt sich bereits in mehr als 100 Betrieben. Besonderen Anteil daran hat der VEB Ingenieurbüro und Rationalisierung Plauen. Wie Direktor Lothar Bühring informierte, fertigt der Kombinatsbetrieb in diesem Jahr „dekotronik"-Erzeugnisse im Wert von vier Millionen Mark. Als mikroelektronisches Zentrum des Kombinates Deko werde er weiter ausgebaut und profiliert. Zu den 800 Webmaschinen, die bereits mit „dekotronik"-Anlagen ausgerüstet sind, kommen weitere hinzu. Gegenwärtig werden 23 Wellenfachwebmaschinen im VEB Vowetex Plauen damit ausgestattet. Auch die Kombinate Baumwolle – hier wurden 300 Webmaschinen mit „dekotronik" modernisiert –, Technische Textilien, Trikotagen und das ESDA-Strumpfkombinat setzen verstärkt auf diese elektronische Erfassung von Daten zur rationellen Gestaltung von Produktionsprozessen. Damit werde, so Lothar Bühring, eine effektive Flächenwirkung erreicht. Jetzt richte sich die Aufmerksamkeit auf Aufgaben zur flexiblen Automatisierung ganzer Produktionsprozesse. *ADN*

Interessengemeinschaft Rechentechnik gegründet

Die effektive Auslastung von Bürocomputern und das Nachnutzen vorhandener Programme ist das Ziel einer jetzt im Kreis Sternberg gegründeten Interessengemeinschaft Rechentechnik. Spezialisten aus neun Betrieben finden sich zu diesem Zweck regelmäßig zusammen. Erstes konkretes Ziel ist die Erarbeitung eines Programms zur Materialwirtschaft, das die Betriebe auch auf verschiedenen Computertypen anwenden können.

Basis für die Gründung der Interessengemeinschaft war eine gute Bilanz der seit 1987 bestehenden Kooperationsgemeinschaft „Rationalisierungshilfe". Im Vorjahr schufen die sieben daran beteiligten Partner Rationalisierungsmittel im Wert von 600 000 Mark. *ADN*

Computer- und Softwaremesse in Wien

Neue Entwicklungen im Bereich der Büro- und Kommunikationstechnik wurden auf der 20. IFABO-Fachmesse und der angeschlossenen Software-Messe PROGRAMMA präsentiert, die Ende April für fünf Tage ihre Tore in Wien öffneten. Mit 565 Direktausstellern aus elf Ländern, die auf fast 50 000 Quadratmetern Fläche insgesamt 1 364 Firmen aus 31 Ländern vertraten, verzeichnete die Exposition einen Teilnahmerekord. Am Stand des DDR-Außenhandelsbetriebes Robotron Export-Import wurden elektronische Schreibmaschinen, Zeichentechnik, Software und der 16-Bit-Personalcomputer EC 1834 gezeigt. Das gewachsene Interesse an dieser Messe entspricht der zunehmenden Bedeutung der Informatik-Industrie für die österreichische Wirtschaft. Der Umsatz von Produkten und Dienstleistungen dieser Branche, in der mehr als 60 000 Beschäftigte arbeiten, erreichte im vergangenen Jahr über 30 Milliarden Schilling. Das sind zwei Prozent des Bruttoinlandsproduktes. Mit 74 Prozent ist der Anteil österreichischer Wertschöpfung bei Software besonders hoch. *ADN*

Schweriner Softwaremesse

50 ausgewählte Programme, vorrangig für Arbeitsplatz- und Personalcomputer, wurden Anfang Mai auf der zweiten Schweriner Softwaremesse zur Nachnutzung angeboten. Die Veranstalter – der Bezirksverband Schwerin der Kammer der Technik, die Bezirksfachsektion Rechnergestützte Arbeit/Mikroelektronik und die Betriebssektion des Datenverarbeitungszentrums Schwerin – übergaben den rund 400 Interessenten einen übersichtlichen Softwarekatalog. Darin offerierten 16 Betriebe und Einrichtungen Schwerins und anderer Bezirke nachnutzbare Rationalisierungslösungen zum Beispiel für Projektierung und Materialwirtschaft. Sie waren in Fachvorträgen erläutert und anschließend an 13 Arbeitsstationen vorgeführt worden.

Das Datenverarbeitungszentrum Schwerin stellte zum Beispiel ein System vor, das Büro- und Personalcomputern den Zugriff zu ESER-Datenbanken erlaubt. Potentielle Nachnutzer konnten auf der Messe Disketten sofort überspielen. Zudem bestand die Möglichkeit, Hardware zu tauschen. *ADN*

USA drohen Japan mit „neuem Herangehen" an bilateralen Handel

Die USA haben Japan vorgeworfen, das 1986 zwischen beiden Staaten abgeschlossene Abkommen über den Handel mit Halbleitern nicht einzuhalten. Dieses sieht vor, daß amerikanischen Firmen bis 1991 ein Anteil von 20 Prozent am japanischen Chip-Markt eingeräumt wird. Wie die US-Handelsbeauftragte C. Hills vor dem Finanzausschuß des Senats erklärte, sei der vor drei Jahren 8,5 Prozent betragende US-Anteil durch mangelndes japanisches Entgegenkommen bis jetzt nur auf 10,7 Prozent gestiegen. Jetzt würden die USA ein neues Herangehen an den Handel mit Japan erwägen.

Um den im vergangenen Jahr 52 Milliarden Dollar betragenden amerikanischen Importüberschuß – das sind 44 Prozent des gesamten USA-Außenhandelsdefizits – abzubauen, solle Japan künftig feste Vorgaben für Einkäufe in den Vereinigten Staaten erhalten. Die Politikerin gestand ein, daß das eine vollkommene Abkehr von der bisher von der Administration propagierten „Freihandelspolitik" darstellt. *ADN*

Neue Runde im „Chip-Krieg" Japan – USA

Japan geht im „Chip-Krieg" mit den USA in die Offensive. Ende April wurde von der Regierung in Tokio der Bann über ein bisher streng vertrauliches Papier aufgehoben, das die Vereinigten Staaten als ihre schärfste Waffe in dem seit Jahren latenten Streit um Anteile auf dem japanischen Halbleitermarkt benutzen. Die Nachrichtenagentur Kyodo veröffentlichte den Wortlaut eines detaillierten Dokumentes aus dem Jahre 1986, das damals dem recht allgemein gehaltenen bilateralen Chip-Abkommen beigefügt wurde und aus dem die USA-Halbleiterindustrie seither ableitet, ab 1990 ein Recht auf 20 Prozent des japanischen Marktes zu haben. Mit der Veröffentlichung des Dokumentes – geschrieben vom Botschafter Japans in Washington an den damaligen Handelsbeauftragten Yeutter – weist das Ministerium für Internationalen Handel und Industrie in Tokio diese Absicht der USA als Wunschdenken zurück. Zwar schreibe der Diplomat, daß Japans Regierung eine Verdopplung des USA-Anteils am Chip-Markt für „realisierbar und wünschenswert" halte, aber verpflichtet habe man sich zu überhaupt nichts.

Ob sich Marktanteile sichern lassen, hänge von der technologischen Kraft der Anbieter, deren Verkaufsstrategie und Preisen sowie dem eingesetzten Kapital ab und nicht von politisch motivierten Forderungen, argumentiert das Ministerium. Das offizielle Tokio reagiert damit zum ersten Mal deutlich auf die Versuche amerikanischer Elektronikkonzerne, eine Verdopplung des Marktanteils in Japan von derzeit etwa zehn Prozent durch weitere Handelsrestriktionen seitens der USA-Regierung gegen japanische Waren durchzusetzen. Bereits 1987 hatte Washington zur Abwehr japanischer Konkurrenz Schutzzölle gegen elektronische Produkte des fernöstlichen Industriestaates eingeführt. Auch wenn diese protektionistische Maßnahme später gelockert wurde, entsteht den Japanern daraus noch heute ein Schaden von jährlich 165 Millionen Dollar.

Japans Elektronikriesen haben in den letzten Jahren ihre Kapazitäten derart schnell erweitert, daß sie nicht nur den Binnenmarkt zu 88 Prozent selbst abdecken, sondern darüber hinaus in großem Umfange hochintegrierte Schaltkreise exportieren.

ADN – Köhler

Siemens und Iskra gründen Joint-Venture

Der jugoslawische Elektrokonzern Iskra und die Siemens AG haben im Mai in Ljubljana einen Vertrag zur Gründung eines Joint-Ventures geschlossen, an dem Siemens zu 48 Prozent und Iskra zu 52 Prozent beteiligt sein werden. Für Siemens hat Vorstandsmitglied Erwin Hardt, für Iskra deren Präsident Franc Sifković unterzeichnet. Das neue Unternehmen firmiert unter dem Namen Iskratel.

Das Joint-Venture hat die Aufgabe, das öffentliche Fernsprech-Vermittlungssystem EWSD von Siemens in Jugoslawien einzuführen. In die Kooperation wird außerdem das von Iskra entwickelte Vermittlungssystem SI 2000 eingebracht, das für kleine bis mittlere Anlagen konzipiert ist. Dieses System wird auch als Nebenstellenanlage eingesetzt. Mit beiden Vermittlungssystemen soll das jugoslawische Fernsprechnetz weiter digitalisiert und ausgebaut werden. Es ist darüber hinaus der Export beider Systeme in andere Länder vorgesehen. *MP*

Kopplung von UNIX-Rechnern

Detlef Gierth
Kombinat VEB Elektro-Apparate-Werke „Friedrich Ebert" Berlin-Treptow
Zentrum für Forschung und Technologie
Uwe Inhoff
VEB Robotron-Projekt Dresden

Einführung

Mit MUTOS, WEGA und VMX wird für alle in der DDR produzierten 16- und 32-Bit-Rechner sowie für ESER-Rechner eine weitgehend einheitliche Betriebssystemumgebung bereitgestellt, die die durch UNIX standardisierten Schnittstellen realisiert. Der Leistungsumfang dieses Betriebssystems ist in /1/ und /2/ beschrieben.

Mit der schnellen Verbreitung dieses Systems entsteht auch die Forderung nach Kopplung der verschiedenen Rechner, auf denen diese Betriebssysteme laufen. Diese Forderung basiert auf folgenden Punkten:

① Es muß der Zugriff auf gemeinsam geführte Dateien und Datenbestände ermöglicht werden.

② Der Zugriff auf teure Peripherie (z. B. schnelle Drucker, Plotter, ...) soll auch von anderen Anlagen möglich sein.

③ Der Datenaustausch mit transportablen Datenträgern (z. B. Disketten) nimmt viel Zeit in Anspruch, und an den verschiedenen Geräten werden nicht immer die gleichen Datenträger unterstützt.

Aufgrund der Kompatibilität der Betriebssysteme ist ein Datenaustausch zwischen verschiedenen Rechnern problemlos möglich. Eine für den Nutzer leicht zu handhabende Lösung der Online-Kopplung, die aber gleichzeitig eine sehr hohe Datensicherheit gewährleistet, bietet das Programmpaket UUCP /3/, /4/.

Eine ausführliche Beschreibung zur Bedienung, Funktionsweise und Implementierung von UUCP ist in der Dokumentation der UUCP-Komponente für das jeweilige Betriebssystem enthalten. Im folgenden werden die Besonderheiten beschrieben, die bei der Kopplung von Rechnern verschiedenen Typs zu beachten sind. Der vorliegende Beitrag soll Anwendern, die sich mit Hilfe der entsprechenden Dokumentation bereits mit dem Programmpaket UUCP bekannt gemacht haben, helfen, heterogene Rechnernetze aufzubauen.

Das Programmpaket UUCP

UUCP ermöglicht den Zusammenschluß von verschiedenen Rechnern mit einem UNIX-kompatiblen Betriebssystem zu einem Netz. In diesem Netz ist die in der Handhabung vom Rechnertyp unabhängige, gesicherte Übertragung beliebiger Dateien sowie die entfernte Abarbeitung von Kommandos möglich.

Zur Zeit können folgende Rechner mit UUCP vernetzt werden:

Rechnertyp	Betriebssystem
K 1840	MUTOS 1800
K 1630	MUTOS 1630
P 8000	WEGA
A 7100/7150	MUTOS 1700
EC 1834	MUTOS 1834

Für den Anschluß an das Netz wird keine spezielle Hardware benötigt. Die Kopplung erfolgt über Terminalschnittstellen, die auch zur Bedienung der lokalen Terminals benutzt werden. Daraus resultieren die für die Kopplung benutzbaren Übertragungsstrecken, die in Tafel 1 aufgeführt sind.

Tafel 1 Mögliche Koppelstrecken für UUCP

Interface	Übertragungsgeschwindigkeit (in Baud)	Übertragungsentfernung
IFFS	maximal 9600	bis 500 m
V.24 direkt	maximal 9600	bis 15 m
V.24 mit Datennahübertragungseinrichtung (DNÜ K 8172) (Vierdraht)	1200 · · · 9600	bis 30 km · · · bis 10 km
V.24 mit Modem	maximal 2400	unbegrenzt

Alle Rechner des Netzes sind berechtigt, zu jedem beliebigen Zeitpunkt eine Verbindung zu einem oder mehreren Partnern aufzubauen. Der Datenaustausch kann dabei zwischen miteinander direkt verbundenen Rechnern sowie über Netzknoten hinweg erfolgen.

Kopplung von Rechnern verschiedenen Typs

Die UUCP-Implementierungen auf den verschiedenen Rechnern sind hinsichtlich des Übertragungsprotokolls und des Nutzerinfaces zueinander kompatibel. Für den Nutzer von UUCP ist die Handhabung unabhängig vom Typ des Partnerrechners.

Unterschiede zwischen den verschiedenen Rechnern bestehen in der konkreten Ausführung des physischen Leitungsanschlusses und in der Codierung des Anfangszeichens für den Anmeldevorgang am System. Diese Unterschiede müssen beim Konfigurieren eines Netzes beachtet werden, haben aber für die spätere Nutzung keine Auswirkungen.

Physische Verbindung

IFSS-Verbindung

Mit der stromgesteuerten, seriellen Schnittstelle IFSS können Entfernungen von bis zu 500 m überbrückt werden. Folgende Leitungen werden für die IFSS-Verbindung benötigt:

- Sendedaten – (SD–)
- Sendedaten + (SD+)
- Empfangsdaten – (ED–)
- Empfangsdaten + (ED+)

Die Leitungen zweier Rechner sind folgendermaßen zu verbinden:

SD– ——— ED+
SD+ ——— ED–
ED– ——— SD+
ED+ ——— SD–

In Tafel 2 sind die Steckerbelegungen für aktive Sender und passive Empfänger dargestellt.

V.24-Direktverbindung

Die spannungsgesteuerte, serielle Schnittstelle V.24 läßt Entfernungen bis ca. 15 m zu. Für die Kopplung von Rechnern über ein Direktverbindungskabel werden folgende Leitungen der V.24-Schnittstelle benötigt:

	Bezeichnung nach CCITT
Betriebserde (SG)	102
Empfangsdaten (R × D)	103
Sendedaten (T × D)	104

Die Leitung *Empfangsdaten* des einen Rechners ist mit der Leitung *Sendedaten* des anderen Rechners zu verbinden. Der Schirm ist einseitig an die Betriebserde anzuschließen.

Verbindung mit Datenübertragungstechnik

Zur Überbrückung größerer Entfernungen müssen in die Übertragungsstrecke Datenübertragungseinrichtungen (DÜE) eingeschaltet werden. In Tafel 1 sind die damit möglichen Übertragungsentfernungen und -geschwindigkeiten aufgeführt.

Bei der Auswahl der Übertragungseinrichtungen ist zu beachten, daß die Übertragung ausschließlich asynchron und im Duplexbetrieb erfolgt. Dementsprechend werden auch nur die für den Asynchron-Duplexmodus benötigten Steuer- und Meldeleitungen der V.24-Schnittstelle bedient.

Alle in Tafel 4 enthaltenen Rechnertypen

Tafel 2 Steckerbelegungen der IFSS-Verbindung

	A 7100/7150 und K 1840	K 1630	EC 1834	P 8000
Stecker	Cannon 25polig	EFS 5polig	Cannon 15polig	Cannon 25polig
Pinbelegung	ED+ –14 ED– –13 SD+ –19 SD– –10 Schirm –1	ED+ –A3 ED– –B4 SD+ –B2 SD– –A1 Schirm – A5	ED+ –12 ED– –15 SD+ – 9 SD– –10 Schirm –1	ED+ –13 ED– –14 SD+ –10 SD– –19 Schirm –7 Brücke 7–9 innerhalb des Steckers
Bemerkungen		Kanal 1A nicht benutzen (bei Standardeinstellung im Taktverteilungsfeld ist Übertragungstakt fest)		Ab Versionsnummer V.43xx (Produktionsbeginn 3/89): ED+ –14 ED– –13 SD+ –19 SD– – 7 Schirm –7 Brücken: 12–10 9– 7

Tafel 3 Steckerbelegungen der V.24-Direktverbindungen

	A 7100/7150 und K 1840	K 1630	EC 1834	P 8000
Stecker	Cannon 25polig	EFS 26polig für 2 Kanäle	Cannon 15polig	Cannon 25polig
Pinbelegung	R × D–3 T × D–2 SG –7	R × D–B04 T × D–A03 SG –A01 für Kanal A	R × D–3 T × D–2 SG –7	R × D–2 T × D–3 SG –7
Bemerkung	Erfolgt beim K 1840 eine V.24 Direktverbindung über einen für Modembetrieb vorgesehenen AMF-Kanal, muß im Stecker die Nullmodemfunktion realisiert werden.	Kanal 2A benutzen; Takt T2 statt T2H einspeisen (Brücke X1701 – X1705)		ab Versionsnummer V.43xx: R × D–3 T × D–2 SG–7

können mittels Datenübertragungstechnik miteinander gekoppelt werden. Der Umfang der Schnittstellensteuerung ist jedoch von Rechner zu Rechner verschieden. Eine vollständige Modemsteuerung mit Überwachung der Meldeleitungen der DÜE existiert nur für den K 1840. Bei den Rechnern A 7100/7150 und EC 1834 ist ein automatisches Bereitmachen der DÜE (innerhalb der Funktion open) enthalten. Die Meldeleitungen werden jedoch nicht ausgewertet. Am K 1630 werden die Steuerleitungen nicht bedient. Dort muß die Bereitschaft der DÜE mittels einer Brücke im Steckverbinder (siehe Tafel 4) fest eingestellt werden.
Diese Unterschiede in der Bedienung der Schnittstellenleitungen haben jedoch keinen Einfluß auf die Nutzung der Kopplung. Selbst bei Nichtbeachtung der Meldeleitungen der DÜE gewährleistet das in UUCP enthaltene Datensicherungsprotokoll eine gesicherte Übertragung der Daten.

Logische Netzkonfiguration

Eine Beschreibung der aktuellen Konfiguration des Netzes ist an jedem Rechner in einigen von UUCP genutzten Dateien (L.sys, USERFILE, SQFILE) abgelegt. Diese Dateien haben Textformat und müssen bei der Installierung von UUCP eingerichtet werden. Ausführliche Beschreibungen dieser Dateien sind in den UUCP-Dokumentationen für die einzelnen Betriebssysteme enthalten.
Bei der Kopplung von Rechnern verschiedenen Typs ist die unterschiedliche Codierung des Anfangszeichens zum Anmelden am System sowie die möglicherweise andere Übertragungsgeschwindigkeit zu beachten. Das Format der oben erwähnten Konfigurationsdateien ist für alle UUCP-Systeme gleich. Der einzige Unterschied besteht in der Darstellungsart des Anfangszeichens zum Anmelden am Partnerrechner.
In Tafel 5 sind Beispiele für die Einträge in der Datei L.sys aufgeführt, die diese Darstellungsarten demonstrieren. Als Namen für die Partnersysteme wurden hier die Bezeichnungen der einzelnen Rechnertypen gewählt. Die Einträge für die Rechner A 7100/7150, K 1630 und EC 1834 sind identisch und deshalb nur einmal aufgeführt.
Die Zeichen ˆD und ˆZ wurden hier für die Darstellung der Codierung 004 bzw. 032 (oktal) benutzt.
Im Betriebssystem WEGA muß jeder Nutzereintrag in der Paßwortdatei ein Paßwort enthalten (auch der Eintrag für UUCP). In den MUTOS-Systemen ist die Spezifikation eines Paßwortes optional.

Kopplung von MUTOS/WEGA mit anderen UNIX-kompatiblen Systemen

Mit Hilfe des Programmpaketes UUCP ist die Kopplung von MUTOS und WEGA mit anderen Rechnern mit UNIX-kompatiblen Betriebssystemen (z. B. Personalcomputer unter XENIX, SINIX, ...) möglich, wenn dort die UUCP-Komponenten vorhanden sind. Hierbei ist jedoch zu beachten, daß bei den meisten UUCP-Implementierungen die Verbindungsaufnahme über eine Terminalleitung nur in einer Richtung möglich ist, das heißt, der eine Partner übernimmt die MASTER-Rolle, während der andere, passive Partner, als SLAVE fungiert. Soll die Verbindungsaufnahme von beiden Seiten erlaubt werden, müssen für die Kopplung 2 Leitungen benutzt werden. Bei MUTOS und WEGA ist eine gleichberechtigte Verbindungsaufnahme von beiden Seiten über eine Leitung möglich.
Bei einer Kopplung von Rechnern unter MUTOS oder WEGA mit einem anderen Rechner muß diesem Rechner die MASTER- oder die SLAVE-Rolle fest zugeordnet werden. Ist der Partnerrechner aktiv (MASTER), muß dort ein Login-Prozeß für den Kopplungskanal verhindert werden. Bei Zuordnung der SLAVE-Rolle muß ein Login-Prozeß für den Kopplungskanal aktiv sein. Weiterhin wird in der Datei L.sys in der Spalte *time*, die die erlaubte Zeit für eine Verbindungsaufnahme enthält, *Never* eingetragen. Auf der Seite der MUTOS- bzw. WEGA-Rechner sind keine Änderungen notwendig.
Bei der oben beschriebenen Arbeitsweise werden die Aufträge eines am MASTER-Rechner arbeitenden Nutzers sofort ausgeführt, während die am SLAVE-System ausgelösten Aufträge zunächst nur im Spool-Directory gespeichert werden. Die Abarbeitung dieser Aufträge erfolgt erst bei der nächsten, von seiten des MASTERs aufgebauten Verbindung. Dieser Verbindungsaufbau kann auch zyklisch durch einen Daemon-Prozeß veranlaßt werden.

Schlußbemerkungen

Mit dem Programmpaket UUCP ist eine Vernetzung von Rechnern verschiedenen Typs mit einem UNIX-kompatiblen Betriebssystem mit geringem Aufwand und ohne Veränderung der bestehenden Hard- und Software möglich.
Die Bedienerschnittstelle zu UUCP ist einfach, da Funktionen wie die Auswahl der Übertragungsstrecken und -parameter, Zwischenspeicherung zu übertragender Daten oder die Kontrolle der Zugriffsberechtigung zu den Daten intern von UUCP vorgenommen werden. Das von UUCP genutzte Datensicherungsprotokoll unterstützt eine in der Handhabung vom Rechnertyp unabhängige Übertragung beliebiger Daten und ist somit für eine breite Palette von Anwenderlösungen einsetzbar.
Aufgrund des für die Kopplung benutzten Interfaces ist die Datenübertragungsrate begrenzt. Die Nettodatenraten liegen bei 200...500 Zeichen/s. Für Anwendungen, bei denen größere Datenmengen in sehr kurzer Zeit übertragen werden müssen, ist UUCP nicht geeignet. Dort müssen ein schnelleres Übertragungsmedium (LAN) und eine im Betriebssystemkern integrierte Netzkomponente zum Einsatz kommen. Gegenwärtig wird im VEB Robotron-Projekt Dresden an der Unterstützung des ROLANET2 im Betriebssystem MUTOS 1800 gearbeitet.

Tafel 4 Steckerbelegungen für den Anschluß von Datenübertragungstechnik

	A7100/7150	K 1630	K 1840	EC 1834	P 8000
Stecker	Cannon 25polig	EFS 26polig für 2 Kanäle	Cannon 25polig	Cannon 5polig	Cannon 25polig
Pinbelegung	T × D (103) −2 R × D (104) −3 RTS (105) −4 CTS (106) −5 DSR (107) −6 DTR (108) −20 DCD (109) −8 DSRS (111) −23 SG (102) −7 PG (101) −1	T × D −A03 R × D −B04 RTS −A05 CTS −B06 DSR −A07 DTR −B08 DCD −A09 DSRS −B10 SG −A01 PG −B01 für Kanal A	T × D −2 R × D −3 RTS −4 CTS −5 DSR −6 DTR −20 DCD −8 DSRS −23 SG −7	T × D −2 R × D −3 RTS −4 CTS −5 DSR −6 DTR −13 DCD −8 DSRS −12 SG −7 PG −1	T × D −2 R × D −3 RTS −4 CTS −5 DSR −6 DTR −20 DCD −8 SG −7
Bemerkungen	in Klammern Bezeichnung nach CCITT; DSRS auf ASP ein; DTR, RTS mit „open" ein	Kanal 2A benutzen; Takt T2 statt T2H einspeisen (Brücke X1701 − X1705); DSRS über AIS ein; RTS, DTR über Brücke im Steckverbinder ein: A05-B08-B10	nur die ersten beiden Kanäle jeder AMF für Modembetrieb (z. B. ttyh0 und ttyh1 am 1. AMF); RTS, DTR, DSRS mit „open" ein; DCD überwacht	RTS, DTR, DSRS mit „open" ein	erst ab Versionsnummer V.43xx: nur tty0 und tty4; bei allen anderen Geräten sind Schaltungsänderungen erforderlich; RTS, DTR mit „open" ein

Tafel 5 Beispiele für Einträge in der Datei L.sys

lokaler Rechner	
P 8000	P8000 Any tty7 9600 tty7 "" NL login:-NL-login: uucp Password: Paul K1840 Any tty6 9600 tty6 ""ˆD login:-ˆD-login: uucp K1630 Any tty5 2400 tty5 "" ˆZ login:-ˆZ-login: uucp
K 1840	P8000 Any ttyh2 9600 ttyh2 "" CR login:-CR-login: uucp Password: Paul K1840 Any ttyh0 9600 ttyh0 "" EOT login:-EOT-login: uucp K1630 Any ttyh3 2400 ttyh3 "" CTRLZ login:-CTRLZ-login: uucp
K 1630	P8000 Any tty1 2400 tty1 "" x login:-x-login: uucp Password: Paul K1840 Any tty2 2400 tty2 "" ˆD login:-ˆD-login: uucp K1630 Any tty3 2400 tty3 "" BREAK login:-BREAK-login: uucp

Literatur

/1/ Claßen, L.: Das Betriebssystemkonzept UNIX. Mikroprozessortechnik, Berlin 2 (1988) 8, S. 227
/2/ Claßen, L.; Oefler, U.: Unix und C. Berlin: VEB Verlag Technik 1987
/3/ Damme, L.; Inhoff, U.: Unterstützung der Datenfernverarbeitung im Betriebssystem MUTOS 1630 mit Softwarepaket UUCP. Neue Technik im Büro, Berlin 31 (1987) 1, S. 20
/4/ Dr. Hamann, K.; Inhoff, U.; Schultz, K.-H.: Netzsoftware für das RVS K 1840. Neue Technik im Büro, Berlin 32 (1988) 3, S. 95

✉ KONTAKT ☎

Kombinat VEB Elektro-Apparate-Werke „Friedrich Ebert" Berlin Treptow, Zentrum für Forschung und Technologie, Abt. WAE, Storkower Straße 101, Berlin 1055; Tel. 4 38 85 09

Portabel programmieren mit C

Prof. Dr. Uwe Hübner
Technische Universität Karl-Marx-Stadt, Sektion Informatik

Die Verbreitung einer vorhandenen oder neuen Programmiersprache wird nicht ausschließlich dadurch bestimmt, daß die Sprache selbst bestimmte Vorzüge gegenüber ihren Konkurrenten hat. Ein weiteres Kriterium ist die Lauffähigkeit auf verschiedenen Rechnertypen und Betriebssystemumgebungen. Die Erfüllung dieses Kriteriums wird Portabilität genannt.
Bei der Entwicklung eines Programmiersystems für die Mehrrechnersteuerungen digitaler Vermittlungsanlagen wurden die nachfolgend dargestellten Erfahrungen gewonnen. Die Portabilitätsproblematik hat dabei aus folgenden Gründen eine große Bedeutung:
① Software für große Prozeßsteuerungen wird mit einer Rechentechnik entwickelt, die maximale Produktivität für größere Entwicklerkollektive sichert; es ist jedoch auch notwendig, Softwaremodifikationen vor Ort mit bescheidenerer Technik auszuführen.
② Das Unterstützungssystem stellt einen erheblichen Wert dar, der auch bei einem Wechsel der rechentechnischen Basis (Gerätetechnik, Betriebssystem) erhalten bleiben soll.
Das Programmiersystem beinhaltet unter anderem CHILL-Compiler, Crossassembler (K 1810 WM86) und Testunterstützungen /1/.
Die Unterstützungssoftware wurde in C geschrieben, da diese Sprache an praktisch alle verbreiteten Rechner in guter Qualität angepaßt werden kann.
Im Idealfall könnte man Portabilität so definieren, daß ein beliebig formuliertes Programm auf verschiedenen Implementierungen der Sprache stets identische Resultate liefert. Diese Forderung wird von C nicht erfüllt, da sie im Widerspruch zu der Forderung nach effektiver Nutzung verschiedener Rechnerarchitekturen (unter Kontrolle des Programmierers) steht /2/ /3/.
Die real erreichbare Form der Portabilität besteht jedoch darin, Programme so zu formulieren, daß bei Abarbeitung auf unterschiedlicher Implementierung gleiche Resultate entstehen. Bestimmte Möglichkeiten der Sprache können dabei aber nicht bzw. nur in ganz bestimmten Zusammenhängen verwendet werden /4/ /5/.
Die Erfahrung bei dem erwähnten Softwareprojekt hat gezeigt, daß ohne Beachtung der Portierungsproblematik beim Erstentwurf die Überführung auf eine andere Implementierung je nach Problemkreis und Programmierstil zwischen 10 und 16 Prozent des Aufwands der Erstentwicklung betragen kann (Der minimale Aufwand besteht in einer Neuübersetzung und Überprüfung mittels geeigneter Testdatensätze).
In diesem Beitrag soll dargestellt werden, wie ein minimaler Portierungsaufwand erreicht werden kann. Leider ist das nicht in Form eines *Kochrezepts* möglich, da eine Diskussion der Tücken einzelner C-Implementierungen aus Umfangsgründen und auch wegen der Kurzlebigkeit solcher Informationen nicht sinnvoll erscheint; vielmehr wird versucht, die zu beachtenden Aspekte dazustellen und Lösungsvorschläge vorzustellen, die von den konkreten Eigenschaften möglichst wenig abhängig sind.
Von der Portabilität zu unterscheiden ist das Problem unterschiedlicher Zielumgebungen für Cross-Programmiersysteme (Retargetability); was aber nicht Gegenstand dieses Beitrags sein soll.

Einschränkungen

Schwierigkeiten beim Übergang auf eine andere C-Umgebung (auch auf dem gleichen Prozessor) können daraus entstehen, daß es (meist geringfügige) Abweichungen der zulässigen Syntax oder Lexik der Sprache gibt:
- Die Anzahl der relevanten Zeichen von Bezeichnern ist in einigen Implementierungen auf 8 begrenzt; mitunter lassen Assembler/Linker noch weniger Zeichen zu!
- C ist eine *lebende Sprache*, daß heißt es können neue Elemente hinzukommen (z. B. enum-Typ, void-Typenangabe); ältere, als nachteilig erkannte Dinge können wegfallen.

In /6/ ist dargestellt, wie die Entwickler der Sprache diese Evolution handhaben. In nächster Zeit zu erwartende Entwicklungen gehen in Richtung eines strengeren Typenkonzeptes (Parametertypenangabe bei externen Prozeduren).
Qualitativ neue Möglichkeiten werden mit C++ /7/ erschlossen.

Speichermodelle

Eine höhere Programmiersprache sollte ja eigentlich den Programmierer von Überlegungen zur physischen Repräsentation und Lage des entstehenden Maschinenprogramms entheben. In Bezug auf die maximale Programmlänge und die Arbeit mit Zeigern ist jedoch eine Beachtung des Speichermodells notwendig.
Das konzeptionell einfachste Modell ist der linear adressierte Speicher, Dies ist im allgemeinen erst für 32-Bit-Rechner praktikabel, da bei 16-Bit-Rechnern mit einer linearen 16-Bit-Adresse nur 64 KByte Speicherraum adressiert werden können. Der am häufigsten verwendete Ausweg ist die Speichersegmentierung, das heißt der Speicher wird in Segmente von jeweils (maximal) 64 KByte eingeteilt; die Segmentlage wird durch Segmentbasisadressen, Segmentnummern oder ähnliches vorgegeben; eine ausführlichere Diskussion dieser Problematik ist in /8/ zu finden.
Drei verbreitete Modelle werden nachfolgend aufgeführt:
Im **kleinen Modell** befinden sich Kode und Daten eines Programms gemeinsam in einem Segment von maximal 64 KByte.
Das **mittlere Modell** stellt je ein Segment zu maximal 64 KByte für Kode und für Daten zur Verfügung (seperated instruction/data); im günstigsten Fall kann ein Programm also 128 KByte belegen.
Beliebig viele Segmente (zu je maximal 64 KByte) stehen im **großen Modell** zur Verfügung; wenn keine virtuelle Speicherorganisation vorhanden ist, begrenzt dann nur der physische Speicherausbau die Programmgröße.
Die Auswahl des Modells muß der Programmierer meist explizit beim Übersetzeraufruf treffen. Nicht immer stehen alle Modelle zur Verfügung (z. B. beim A 5120.16 nur das kleine Modell; bei verschiedenen SKR-Rechnern ohne Overlaysystem nur die Modelle klein und mittel)
Dem kleinen und dem mittleren Modell ist gemeinsam, daß Zeiger als 16-Bit-Werte dargestellt werden; dies führt zu etwas kleineren Programmlängen und kürzeren Laufzeiten als im großen Modell. Leider hat die breite Verfügbarkeit von Rechnern, die ausschließlich diese Speichermodelle besaßen, dazu geführt, daß in Programmen Zeiger in int-Variable überführt werden, um damit Berechnungen anzustellen. Dies sollte unbedingt vermieden werden, da beim großen Modell ein Zeiger nicht mehr in 16 Bit dargestellt werden kann (typisch: 32 Bit).
Ein anderes Problem ist die Addition von int-Werten zu Zeigern oder die Bildung von Differenzen aus Zeigern. Im Sinne des Typenkonzepts von C sind folgende Konstruktionen legal:

```
int m, n;
int *p, *p1, *p2;
p = p + m;
n = p1 - p2;
```

Es sei daran erinnert, daß dies nicht äquivalent zu einer arithmetischen Addition oder Subtraktion ist (m bezeichnet die Anzahl der Datenobjekte vom Typ int, um die p weitergeschaltet werden soll; n die Anzahl der zwischen p1 und p2 liegenden int-Datenobjekte). Es ergibt sich eine Beschränkung der Größe eines einzelnen Datenobjekts (nicht des gesamten Programms) auf den mit einer int-Repräsentation darstellbaren Wertebereich /6/.

Datenrepräsentation

Der Wertebereich der verschiedenen C-Typen für ganze Zahlen ist sehr spezifisch und muß unbedingt der jeweils zutreffenden Beschreibung entnommen werden. Mit Ausnahme einiger Compiler für 8-Bit-Zielprozessoren kann als erster Anhaltspunkt gelten /9/ /10/:

char	8 Bit
short	16 Bit
int	*natürliche* Genauigkeit des jeweiligen Rechners (d. h. 16 Bit für 16-Bit-Rechner usw.)
long	32 Bit

Die vorzeichenlose Darstellung war bei älteren Compilern nur für die Genauigkeit des int-Typs möglich, neuere Versionen lassen unsigned als zusätzliches Attribut für alle Genauigkeiten zu.
Auf eine einheitliche Datenrepräsentation kommt es an, wenn binäre Dateien mit Elementen von mehr als 8 Bit Länge verarbeitet oder erzeugt werden (z. B. Objektmodule) oder eine Arithmetik bestimmter Genauigkeit erforderlich ist (z. B. Konstantenausdrucksberechnung in Compilern bzw. Assemblern). In solchen Fällen ist short bzw. long der Vorzug vor int zu geben.
Leider ist die unterschiedliche Genauigkeit nicht das einzige Problem; schwerwiegender (weil meist nicht so offensichtlich) ist die unterschiedliche Reihenfolge von Elementareinheiten in einer größeren Einheit (Reihenfolge der Bytes im Wort, der Worte im Doppelwort, ...). Da sich für die verschiedenen Varianten jeweils plausible Begründungen finden lassen, sind zum Leidwesen der Anwender auch alle Möglichkeiten in der Praxis vertreten /11/. Für einige gebräuchliche Architekturen gelten die Angaben entsprechend der folgenden Tabelle.

		K 1810 WM86	U 8001
Reihenfolge im Wort	niedrigere Adresse	Low-Byte	High-Byte
	höhere Adresse	High-Byte	Low-Byte
Reihenfolge im Doppelwort	niedrigere Adresse	Low-Wort	High-Wort
	höhere Adresse	High-Wort	Low-Wort

Die portable Lösung solcher Probleme soll an einem Beispiel demonstriert werden. Es sei ein short-Wert (16 Bit) auf ein File auszugeben; als externe Darstellung wird (relativ willkürlich) die erste aus der obigen Tabelle festgelegt (d. h. die natürliche für K 1810 WM86). Eine einfache Lösung könnte sein:

```
short value;
fwrite (&value, sizeof(short), 1, fp_outfile);
```

Diese Lösung würde aber nur bei der K 1810 WM86-Architektur korrekt arbeiten, eine portable Lösung könnte wie folgt aussehen:

```
short value;
putc ((char) value, fp_outfile);
putc ((char) (value/256), fp_outfile);
```

Die Übersichtlichkeit wird verbessert, wenn man für solche Fälle Funktionen einführt (hier z. B. fputshort ...).

Typkonvertierungen

Für die impliziten und expliziten Typkonvertierungen gibt es in C keine festgelegte Semantik. Der kritische Fall ist die Überführung in einen Typ mit größerem Wertebereich, da hier je nach Compiler mit oder ohne Vorzeichenerweiterung gearbeitet wird.
Mit Typumwandlungen sollte aus diesem Grund (aber auch aus Gründen der Übersichtlichkeit) sparsam umgegangen werden, eine explizite Angabe des Konvertierungsoperators macht darauf aufmerksam, daß eine kritische Stelle vorliegt und sollte daher immer erfolgen.
Beachtung verdient auch der Typ des Rückgabewertes von Funktionen. Bei externen Funktionen, die keinen Rückgabewert vom Typ int haben, darf die extern-Deklaration nicht vergessen werden (anderenfalls kann es durch die dann implizit erfolgende Typwandlung zu unerwarteten Ergebnissen kommen).
Die Verwendung von char-Variablen für kleine ganze Zahlen bringt mehr Schwierigkeiten als Nutzen, da viele Rechner mit Datenbreiten >8 Bit solche char-Objekte im Vergleich zu int-Objekten keineswegs mit weniger Programm und Laufzeit bearbeiten. Höchstens bei Feldern ist ein solches Vorgehen zu rechtfertigen:

```
/* Ausschnitt der Kode-/Variantentabelle eines Assemblers */
struct ast {char kode; char variante;};
struct ast astab [ ] = {
0xcc, 3,
0xcc, 5,
...};
int index, ko, var;
ko = ((int) astab [index]. kode) & 0xff;
```

Aber Vorsicht! Konstanten haben im allgemeinen den Typ int. Wenn sie zusammen mit Objekten größeren Wertebereiches verwendet werden, kommt es zu impliziten Typwandlungen, die nicht immer das tun, was der Programmierer erwartet:

```
long bigval;
bigval = bigval & 0x0000ffff; /* fehlerhaft!!! */
```

Vermutlich will der Programmierer den niederwertigen Teil ausblenden; die angegebene Anweisung hat aber in einigen Compilern überhaupt keinen Effekt, da die Konstante als (−1) interpretiert wird und durch die Benutzung in einem long-Ausdruck vorzeichenrichtig auf 0xffffffff erweitert wird. Eine richtige Lösung ist:

```
bigval = bigval & 0xffffl;
```

Bibliotheksfunktionen und Ablaufumgebung

Bei der Verwendung von Bibliotheksfunktionen können Schwierigkeiten daraus entstehen, daß Parametertypen unterschiedlich sind (jedoch seltener), bzw. die physische Repräsentation eines Parameters von Bedeutung ist (siehe oben) oder einige Funktionen nicht überall zur Verfügung stehen; eine gute Hilfe sind hier /12/ /13/.
Eine der Eigenschaften der Ablaufumgebung ist die implizite Initialisierung von statischen Variablen mit einem festen Wert (meist 0). Dies ist aber keine Eigenschaft der Sprache C, Variableninitialisierungen sollten daher dort wo sie notwendig sind zur Unabhängigkeit von der Ablaufumgebung und zur besseren Selbstdokumentierung explizit vorgenommen werden. Bei Programmen mit Dateiarbeit ist die Lage der benutzten Dateien in der Hierarchie zu beachten; kritisch sind hier vor allem solche, die von mehreren Nutzern gemeinsam benötigt werden (z. B. Unterprogrammbibliotheken). Da der genaue Aufbau der Dateisysteme individuell verschieden sein kann, ist es wichtig, daß die entsprechenden Abhängigkeiten nicht im Programm verstreut, sondern in einer Einfügedatei isoliert werden. Für *Notfälle* sollte man überall absolute Pfadnamen zulassen, da sich so auf jeden Fall eine (wenn auch etwas unbequeme) Arbeitsfähigkeit sichern läßt.

Besonderheiten der Rechnerarchitektur

Es ergibt sich die Notwendigkeit einer Ausrichtung von Datenobjekten bzw. Befehlen auf eine gerade Adresse bzw. auf ihre natürliche integrale Grenze (d. h. 2-Byte-Objekte auf durch 2 teilbare Adressen, 4-Byte-Objekte auf durch 4 teilbare Adressen ...)
Die Anzahl und Art der möglichen Registervariablen müssen berücksichtigt werden.
Die Art und Weise der Realisierung des Stackrahmens beim Funktionsaufruf muß beachtet werden; dies kann für Funktionen mit variabler Parameteranzahl von Interesse sein (es sind allerdings auch bei gleichem Prozessor verschiedene Lösungen möglich).
Normalerweise existieren im Übersetzersystem Mechanismen zur Berücksichtigung dieser Besonderheiten; der Programmierer hat aber (durch Typwandlungen) die Möglichkeit, diese Mechanismen zu umgehen und ist dann selbst für die Einhaltung der genannten Aspekte zuständig.

Weiterentwicklung portabler Software

Die Beachtung der Portabilitätskriterien stellt einen Beitrag zur Erhöhung von Qualität (im weiteren Sinne) und Produktivität bei der Softwareentwicklung dar.
Aus Sicht der Softwaretechnologie ist zu empfehlen, bei Wartung und Weiterentwicklung portabler Software von einer Basisversion auszugehen. Eventuell notwendige Spezifika einzelner vorgesehener Ablaufumgebungen werden mit **#ifdef VARIANTE ...** berücksichtigt. Mit dieser Möglichkeit sollte aber sparsam umgegangen werden, da mit jeder neu hinzukommenden Ablaufumgebung dann auch neue Probleme auftauchen.
Nach Modifikationen der Basisversion werden die Quelltexte auf die anderen Ablaufumgebungen gebracht und dort neu übersetzt. Die erneute Prüfung der korrekten Funktion sollte keinesfalls eingespart werden! Der Aufwand dafür ist niedrig, wenn zusammen mit den Quelltexten auch umfassende Testdatensätze und deren Sollresultate überführt werden. Der Vergleich von Soll- und Istresultaten ist sehr einfach (diff).

Literatur

/1/ Hübner, U.: CHILL-Programmiersystem für die Nachrichtentechnik. Nachrichtentechnik Elektronik 38 (1988) 5, S. 188
/2/ Kernighan, B. W.; Ritchie, D. M.: The C Programming Language. Prentice Hall, Englewood Cliffs 1978
/3/ Horn, T.; Kurs: Programmieren in C. Mikroprozessortechnik 1 (1987) 1–6
/4/ Johnson, S. C.; Ritchie , D. M.: Portability of C Programms and the Unix System. Bell Syst. Techn. Journal 57 (1978) 6.2, S. 2021
/5/ Kernighan, B. W.; Pike, R.: The Unix Programming Environment. Prentice Hall, Englewood Cliffs 1984
/6/ Rosler, L.: The Evolution of C – Past and Future. AT & T Bell Labs Techn. Journal 63 (1984) 8.2, S. 1685
/7/ Stroustrup, B.: Data abstraction in C. AT & T Bell Labs Techn. Journal 63 (1984) 8.2, S. 1701
/8/ Furth, B.; Milutinovic, V.: A Survey of Microprozessor Architectures for Memory Management. Computer 20 (1987) 3, S. 48
/9/ Claßen, L.; Oefler, U.: Unix und C. Berlin: Verlag Technik 1987
/10/ Sprachbeschreibung C-Sprache unter den Betriebssystemen MUTOS-8000, MUTOS-1630 VEB Kombinat Robotron 1985
/11/ Bodenstab, D. E.; Houghton, T. F.; Kelleman, K. A.; Ronkin, G.; Schan, E. P.: Unix Operating System Porting Experiences. AT & T Bell Labs Techn. Journal 63 (1984) 8.2, S. 1769
/12/ Bach, F.; Domman, P.: Unix-Tabellenbuch. München: Carl Hanser Verlag 1986
/13/ Rochkind, M. J.: Advanced UNIX Programming. Prentice Hall, Boulder 1985

KONTAKT

Technische Universität Karl-Marx-Stadt, Sektion Informatik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 668430

Fallen in C

Burkhard Lau
Wilhelm-Pieck-Universität Rostock, Sektion Mathematik

Im Rahmen einer intensiven Beschäftigung mit der Systemprogrammiersprache C unter MS-DOS (ohne Verwendung von lint) fällt auf, daß Tücken für Umsteiger von anderen höheren Programmiersprachen auf C in der Standardliteratur /1/, aber auch in /2/ oder /3/ nur ungenügend bzw. nicht dargestellt werden. C wird jedoch mit der Einführung der PCs EC 1834 und A 7150 einen größeren Stellenwert erhalten, wie auch das Programmiersystem C DCP vom LfA beweist /4/. Um deshalb C-Einsteigern Ärger und Mehrarbeit zu ersparen, sei im folgenden auf *Fallen*, die sich aus der C-Philosophie und der Gewöhnung an andere höhere Programmiersprachen ergeben, (nicht gemeint sind Tücken, die auf unterschiedlichen Compilerinterpretationen, wie z. B. i++ = i; oder Bitverschiebungsbefehlen, beruhen!) in diesem Artikel hingewiesen.

Pointerarbeit

Pointer dienen bekanntlich dem indirekten Zugriff auf Objekte jeglichen Typs. Der Definition einer Pointervariablen durch <typ> *pvar; (Anlegen eines 2 bzw. 4 Bytes langen Speicherbereichs im Variablenspeicher) muß jedoch, wie bei Variablen generell, vor der ersten Verwendung eine Initialisierung folgen. Dies kann jedoch nicht durch Eingabefunktionen, sondern nur, falls nicht auf ein schon existentes Objekt gezeigt werden soll, durch die Speicherplatzverwaltungsfunktionen malloc(size) bzw. calloc(n, size) (bei anderen Bibliotheken sicherlich noch andere) geschehen. Sie weisen den Pointern die Anfangsadresse eines freien Speicherbereiches der Länge size im sogenannten Heap vom Typ char zu. Falls ein anderer Typ gefordert wird, so ist eine explizite Typumwandlung durch den Cast-Operatur (<typ> *) durchzuführen. Falsch wäre also: char *pvar; ... gets(pvar);, da undefiniert ist, wohin die durch gets eingegebene Zeichenkette in den Speicher zu schreiben ist. Da mitunter das Betriebssystem betroffen ist, kann es zu mysteriösen Abstürzen kommen. Bestenfalls erhält man auf stderr die Mitteilung: NULL POINTER ASSIGNMENT. Richtig wäre: char *pvar = malloc(30); ... gets(pvar);

Zeichenketten

Leider existiert der Typ String in C nicht, obwohl man doch ständig mit Zeichenketten zu tun hat. Die Simulation von Strings erfolgt entweder über Felder von Zeichen oder über Zeiger auf Zeichen. Doch diese Tatsache ruft neben den allgemeinen Problemen mit Pointern auch noch Probleme in der Verwendung der Stringoperationen hervor. Im folgenden seien a und b so definiert: char *a = malloc(20), *b = malloc(20);

strcpy Eine Verdopplung einer Zeichenkette erreicht man nicht einfach über a = b;. Durch diese Anweisung wird nur Zeiger a auf den Anfang der durch b spezifizierten Zeichenkette gesetzt. Verändert man die Zeichenkette unter dem Namen b, ist sie auch unter a nicht mehr verfügbar. Zum Doppeln dient strcpy(a,b).

strcat Man muß wissen, daß der Compiler alle im Quelltext vorkommenden konstanten Zeichenketten (auch die, die bei printf oder scanf als control dienen!) bei der Compilierung im Konstantenspeicher ablegt. Will man nun weitere Zeichen an eine konstante Zeichenkette anhängen, muß man sie zuerst aus dem Konstantenspeicher in den Heap mit strcpy kopieren, wo man auch für den Platz der nachfolgenden Zeichenkette gesorgt haben muß. Bei Nichtbeachtung dieses Faktes würden nachfolgende Daten überschrieben. Im Fall des Konstantenspeichers würde man sich wundern, daß vielleicht das nachfolgende printf nicht funktioniert oder Unsinn entsteht. Denn printf merkt sich die Adresse im Speicher, wo es sein control abgelegt hatte, und liest dann blindlings den dort stehenden Inhalt. Falsch wäre also: strcat("Hallo",gets(b));, sondern die korrekte Version lautet so: strcat(strcpy(a,"Hallo"),gets(b)); Dies ist auch ein Beispiel, wie mit C kompakter Code erzeugt werden kann. Die bisherigen Betrachtungen galten der Darstellung von Strings mit Hilfe von Pointern. Aber auch die Benutzung von Arrays birgt eine Falle in sich, wie das folgende Programmstück beweist: char a[20]; ... a = "Stringzuweisung"; Die Zuweisung klappt deshalb nicht, da a zwar einen Zeiger auf das Array darstellt, durch die Zuweisung jedoch gezwungen wird, seinen konstanten Wert zu ändern und auf die, im Konstantenspeicher befindliche, Zeichenkette zu zeigen. Man würde also dem Compiler zumuten, daß er die Speicherzelle einer Variablen ändert. Korrekt könnte dies so aussehen: strcpy(a,"Stringzuweisung");

Eingabe vom Terminal

Jede Eingabefunktion überprüft bei ihrem Aufruf den Eingabepuffer. Falls er leer ist, wird er bis zum abschließenden ENTER (/n) zum Beschreiben geöffnet, ansonsten wird der Inhalt des Puffers als Eingabestrom ausgewertet. Häufig zur Eingabe nur eines Zeichens benutzt, aber sehr gefährlich, ist getchar() (und alle damit verwandten Funktionen). Falsch wäre:
char z, *str = malloc(20); ... z = getchar(); ... gets(str);
getchar() öffnet während des Programmlaufs den Eingabepuffer und der Bediener kann nun ein (oder auch mehrere) Zeichen und das abschließende /n eingeben. Der Variablen z wird korrekt das erste Zeichen des Eingabepuffers zugewiesen, doch wird gets keinen leeren Puffer, sondern zumindest /n vorfinden. Somit wird die Eingabe nicht eröffnet und str bekommt die restlichen Zeichen des Terminalpuffers bis zum /n (zuzüglich /0 am Ende) durch gets zugewiesen, da gets alle Zeichen bis zum /n liest. Richtig ist es also, nach dem getchar den Terminalpuffer mit fflush(stdin) zu leeren.
Beim Programmablauf benötigt man des öfteren einen Haltepunkt und möchte dafür ebenfalls getchar nutzen. Dazu ein Realisierungsvorschlag als Makro: #define wait puts ("Fortsetzen mit <ET>");(void)getchar(); fflush(stdin)

Harmlose Fallen

In diesem abschließenden Abschnitt möchte ich eine kurze Zusammenstellung von Stolpersteinen präsentieren, die insbesondere Pascalprogrammierer interessieren werden:

- **der \ (backslash):** Er dient in MS-DOS zur Darstellung von Pfaden. Andererseits dient er in C als Einleitung von Steuerzeichen bei der Ausgabe von Strings. Dementsprechend hätte folgende Anweisung nicht den gewünschten Effekt:
puts("c: \ neudaten \tab1"); Berichtigung: Darstellung des \ durch \\ im auszugebenden String.
- **Vergessenes;:** Das Semikolon dient in C, im Gegensatz zu Pascal, nicht zur Trennung von Anweisungen, sondern es macht einen Ausdruck erst zu einer Anweisung.
- **Vergessene Break-Befehle in switch:** Eine Case-Marke ist in C lediglich ein Einsprungpunkt in den laufende Code. Eine Beendigung der Anweisungsfolge wird nicht durch die nächste case-Marke, sondern durch einen Break-Befehl, der möglicherweise nach der nächsten Case-Marke liegen kann, erreicht.
- **= oder = =?:** In C besitzt jede Anweisung einen Wert. So auch die Zuweisung a = b;, da a den Wert und den Typ von b annimmt. Insofern ist der Ausdruck if (a = b) ... nicht falsch, da in C ein Wert logisch als *false* interpretiert wird, wenn er den Wert 0 besitzt, ansonsten ist er *true*. Es wird jedoch leider überhaupt nicht das erreicht, was der Programmierer eigentlich wollte, nämlich den Vergleich durchzuführen: if (a = = b) ...
- **Vergessene Klammern bei einem Funktionsaufruf:** Es ist nicht falsch, statt des Aufrufs einer Funktion mit einer dem Namen nachgestellten, in Klammern gesetzten Parameterliste die Klammern und die Parameterliste wegzulassen. Dieses liefert dann aber die Adresse der Funktion.
- **Variablenbezeichner:** Im Gegensatz zu Pascal ist die Schreibweise in Groß- und Kleinbuchstaben signifikant, das heißt das var, Var und VAR drei verschiedene Variablen definieren
- **Indizierung mehrdimensionaler Felder:** Betrachten wir einmal folgenden Programmcode: int feld[100,4]; Es soll hier also ein zweidimensionales Feld von Integergrößen definiert werden. Dieses Ziel wird aber nicht erreicht, da es in C möglich ist, mehrere Ausdrücke durch Kommata zu trennen. Es werden dann alle Ausdrücke berechnet und der Wert des Ganzen wird durch den am weitesten rechts stehenden Ausdruck bestimmt. Unser obiger Code entspricht also int feld[4]; Korrekt sieht eine solche Anweisung so aus: int feld[100][4];

Literatur

/1/ Kernigham, W.; Ritchie, M.: The C Programming Language. Bell Laboratories, Murray Hill, New Jersey
/2/ Clauß, M.; Fischer, G.: Programmieren mit C. VEB Verlag Technik, Berlin 1988
/3/ Claßen, L.; Oefler, U.: UNIX und C – Ein Anwenderhandbuch. Berlin: VEB Verlag Technik 1987
/4/ Programmiersystem C DCP. edv-aspekte Berlin: 7 (1988) 3, S. 26

Maus-Anwendung unter Turbo-Pascal

Brigitta Schimpf
Bauakademie der DDR, Institut für Technologie und Mechanisierung

Unter MS-DOS und kompatiblen Betriebssystemen kann eine Maus in Turbo-Pascal-Programmen zur Gestaltung einer anspruchsvollen Dialogführung im Textmode eingesetzt werden. Sie bewährt sich als Hilfsmittel bei der Menüauswahl, beim Rollen, Blättern und Markieren von Daten in Bildschirmfenstern und beim Aufheben von Wartezuständen in Entscheidungspunkten. Dialogprogramme sollten sowohl mit als auch ohne Maus-Unterstützung lauffähig sein. Die in diesem Artikel vorgestellten Prozeduren ermöglichen den Einsatz der Maus in Turbo-Pascal-Programmen (Version 4.0).

Aufruf der Maus-Funktion

Die Maus-Funktionen werden über den DOS-Interrupt 33H angesprochen, wobei durch Register AX die spezielle Funktion ausgewählt wird. Bild 1 zeigt einen Satz von Turbo-Pascal-Prozeduren zum Aufruf der Maus-Funktionen im Textmode, der für viele Anwendungsfälle ausreicht. Diese Prozeduren werden im folgenden erläutert (siehe Bild 1):

```
var reg:registers;

procedure INITMOUSE(var anzbutton:byte; var mouseok:boolean);
begin
  if mem[$0000:$00cc]=0      {interruptvektor $33}
  then mouseok:=false        {keine interruptroutine}
  else begin
    reg.ax:=0; reg.bx:=0; intr($33,reg);
    if reg.ax<>0 then mouseok:=true else mouseok:=false;
    anzbutton:=reg.bl;
  end;
end;  {initmouse}


procedure SOFTTEXTMOUSE(screenmask,cursormask:word);
begin
  reg.ax:=10; reg.bx:=0;
  reg.cx:=screenmask; reg.dx:=cursormask;
  intr($33,reg);
end;  {softtextmouse}


procedure SHOWMOUSE;
begin reg.ax:=1;  intr($33,reg);end; {showmouse}


procedure HIDEMOUSE;
begin reg.ax:=2; intr($33,reg); end; {hidemouse}


procedure GETMOUSEPOSITION(var button,x,y:byte);
begin
  reg.ax:=3; intr($33,reg); button:=reg.bl;
  x:=succ(reg.cx div 8); y:=succ(reg.dx div 8);
end;  {getmouseposition}


procedure SETMOUSEPOSITION(x,y:byte);
begin
  reg.ax:=4;  reg.cx:=pred(x)*8;  reg.dx:=pred(y)*8;
  intr($33,reg);
end;  {setmousepositon}


procedure MOUSEWINDOW(x1,y1,x2,y2:byte);
begin
  reg.ax:=7;  reg.cx:=pred(x1)*8;  reg.dx:=pred(x2)*8;
  intr($33,reg);
  reg.ax:=8;  reg.cx:=pred(y1)*8;  reg.dx:=pred(y2)*8;
  intr($33,reg);
end;  {mousewindow}
```

Bild 1
Prozeduren zum Aufruf der Maus-Funktion

Die Prozedur **INITMOUSE** initialisiert die Maus. Dem aufrufenden Programm wird der Erfolg der Initialisierung (mouseok) und die Anzahl der verfügbaren Maus-Tasten (anzbutton) übergeben.

SOFTTEXTMOUSE definiert das Aussehen des Maus-Cursors auf dem Bildschirm. Ein Software-Maus-Cursor wird durch einen Zeichencode und ein Zeichenattribut repräsentiert, die die Bildschirmstelle ersetzen oder verändern, auf der die Maus positioniert ist. Der Wert *screenmask* wird der Prozedur übergeben und legt fest, welche Bildschirmcharakteristika auf der Maus-Position erhalten bleiben sollen. Er wird mit dem aktuellen Bildschirmzeichen und -attribut über AND verknüpft.

Screenmask=80FFH bedeutet beispielsweise, daß das Bildschirmzeichen in seiner Gestalt unverändert bleiben soll (FFH); vom Bildschirmattribut wird jedoch nur die Eigenschaft *Blinken* erhalten, nicht aber Vorder- und Hintergundfarbe (80H). Der Wert cursormask bestimmt die Merkmale, die durch den Maus-Cursor verändert werden sollen. Er wird mit dem Ergebnis der vorherigen AND-Operation über XOR verknüpft. Cursormask=2000H legt für den Maus-Cursor grünen Hintergrund und schwarzen Vordergrund fest (20H); das Bildschirmzeichen wird in seiner Gestalt belassen. Der Aufruf SOFTTEXTMOUSE($80FF,$2000) definiert also den Maus-Cursor als grünen Block, in dem das aktuelle Zeichen schwarz erscheint.

Mit der Initialisierung der Maus und der Auswahl des Maus-Cursors ist die Maus aktiv. Der Maus-Cursor wird jedoch erst nach dem Aufruf der Prozedur **SHOWMOUSE** auf dem Bildschirm sichtbar. Die Prozedur **HIDEMOUSE** läßt dagegen den Maus-Cursor vom Bildschirm verschwinden. SHOWMOUSE und HIDEMOUSE sind stets im Wechsel zu verwenden.

Die augenblickliche Position des Maus-Cursors wird mit Hilfe von **GETMOUSEPOSITION** bestimmt. Diese Prozedur vermittelt auch die Maus-Taste, die zum Zeitpunkt der Abfrage gedrückt war. Dabei bedeuten gesetzte Bits 0, 1 bzw. 2 in der Variablen button, daß die linke, rechte bzw. mittlere Maus-Taste gedrückt wurde.

SETMOUSEPOSITON setzt den Maus-Cursor auf eine bestimmte Bildschirmstelle.

MOUSEWINDOW ist eine Prozedur, die den Maus-Cursor in ein vorgegebenes Fenster zwingt. Hier sind gleiche Werte für die Spalten- bzw. Zeilenangaben möglich, beispielsweise die Beschränkung der Maus-Bewegung auf eine bestimmte Zeile.

Beispiele

An zwei einfachen Beispielen soll die Anwendung der Maus-Prozeduren demonstriert werden:

Die im Bild 2 gezeigte Prozedur **WAIT** erzeugt einen Haltepunkt im Programm, der nach beliebigem Tastendruck auf Tastatur oder Maus beendet wird.

```
procedure WAIT;
var   anzbutton,x,y,button:byte; mouseok:boolean; rk:char;

begin
  INITMOUSE(anzbutton,mouseok);  {Maus Initialisierung}
  if mouseok then begin
     SOFTTEXTMOUSE($80ff,$2000);
     {Maus in das aktuelle Fenster zwingen}
     MOUSEWINDOW(lo(windmin)+1,hi(windmin)+1,
                 lo(windmax)+1,hi(windmax)+1);
     SHOWMOUSE;
  end;
  button:=0;
  while KEYPRESSED do rk:=READKEY;  {Tastaturpuffer leeren}
  repeat
    if mouseok then GETMOUSEPOSITION(button,x,y);
  until KEYPRESSED or (button<>0);
  while KEYPRESSED do rk:=READKEY;  {Tastaturpuffer leeren}
  if mouseok then begin
     HIDEMOUSE; MOUSEWINDOW(1,1,80,25);
  end;
end;  {wait}
```

Bild 2 Beispiel für die Maus-Anwendung Haltepunkt im Programm

Mit Hilfe der in Bild 3 dargestellten Prozedur **DECIDE** wird eine Entscheidung durch eine einzelne Taste gefällt. Zur Entscheidung zugelassen sind dabei alle Tasten, die der Prozedur über den String *Ende Tasten* übergeben werden. In diesem String werden vom aufrufenden Programm je zwei Zeichen für eine Taste bereitgestellt. Das erste Zeichen enthält das Hauptbyte (den ASCII-Code), für Funktions- und Cursor-Tasten den Wert Null. Das zweite Zeichen ist für normale Tasten mit Null, für Funktions- und Cursortasten mit dem Hilfsbyte zu belegen. Ist die Maus installiert, so werden den ersten drei Endetasten die rechte, linke bzw. mittlere Maus-Taste zugeordnet. Eine Maus-Taste kann für die Prozedur DECIDE gesperrt werden, indem die entsprechende Endetaste mit Null (chr(0) +chr(0)) belegt wird.

Beispiele für den Aufruf von DECIDE:

```
var endetasten:string; taste:word;
...
endetasten:=chr($0d)+chr(0)+
            chr($1b)+chr(0)+
            chr(0)  +chr(3b);
DECIDE(endetasten,taste);
case Taste of
    $0d  :...
    $1b  :...
    $3b00:...
end;
...
```

```
procedure DECIDE(var endetasten;var taste:word);
const leftbutton=1; rightbutton=2; middlebutton=4;
var   anztasten,anzbutton,x,y,button:byte; j:integer;
      mouseok,mouse,keybord,ok:boolean; rk:char;

function KEYSTATUS:word;
begin reg.ah:=1; reg.al:=0; intr($16,reg);
      if reg.flags and fzero=0 then begin
         if reg.al<>0 then reg.ah:=0; KEYSTATUS:=reg.ax;
      end
      else KEYSTATUS:=0;
end;  {keystatus}

begin
  INITMOUSE(anzbutton,mouseok);  {Maus Initialisierung}
  if mouseok then begin
     SOFTTEXTMOUSE($80ff,$2000);
     MOUSEWINDOW(lo(windmin)+1,hi(windmin)+1,
                 lo(windmax)+1,hi(windmax)+1);
     SHOWMOUSE;
  end;
  anztasten:=mem[seg(endetasten):ofs(endetasten)] div 2;
  ok:=false;

  repeat        {bis Endetaste}
    while KEYPRESSED do rk:=READKEY;
    keybord:=false;mouse:=false;button:=0;
    repeat     {bis beliebige Taste}
      taste:=KEYSTATUS;
      if (taste<>0) then keybord:=true
      else if mouseok then begin
              GETMOUSEPOSITION(button,x,y);
              if button<>0 then mouse:=true;
           end;
    until keybord or mouse;

    if mouse then begin
      taste:=0;
      if (button and rightbutton=rightbutton) then
           taste:=memw[seg(endetasten):ofs(endetasten)+1]
      else if (button and leftbutton=leftbutton) and
              (anztasten>=2) then
              taste:=memw[seg(endetasten):ofs(endetasten)+3]
      else if (button and middlebutton=middlebutton) and
              (anztasten>=3) then
              taste:=memw[seg(endetasten):ofs(endetasten)+5];
      if taste<>0 then ok:=true;
    end  {if mouse}
    else for j:=1 to anztasten do  {if keybord}
      if memw[seg(endetasten):ofs(endetasten)+2*j-1]=taste
         then ok:=true;
  until ok;
  while KEYPRESSED do rk:=READKEY;
  if mouseok then begin
     HIDEMOUSE; MOUSEWINDOW(1,1,80,25);
  end;
end;  {decide}
```

Bild 3 Beispiel für die Maus-Anwendung, Entscheidungspunkt im Programm

Nach dem Aufruf von DECIDE wird in diesem Beispiel die Eingabe einer der Tasten ENTER, ESC bzw. F1 erwartet, wobei der Druck der rechten Maus-Taste wie ENTER, der linken wie ESC und der mittleren (falls vorhanden) wie F1 wirkt.
Zur Abfrage der Tastatur wird in DECIDE die Funktion KEYSTATUS verwendet, die den Tastencode in der beschriebenen günstigen Weise aufbereitet.
Mit den vorgestellten Hilfsmitteln lassen sich anspruchsvolle Dialogprogramme entwickeln. Die Maus-Tasten sollten dabei stets gleiche Bedeutungen besitzen, z. B.

rechte Taste – bestätigen/auswählen
linke Taste – verneinen/beenden.

Der mittleren Maus-Taste könnten besondere Funktionen zugeordnet werden, beispielsweise bei der Fensterarbeit in Abhängigkeit von der Maus-Position das Rollen der Fensterdaten nach oben bzw. nach unten.

✉ KONTAKT ☏

Bauakademie der DDR, Institut für Technologie und Mechanisierung, Plauener Str. 163–165, Berlin, 1092; Tel. 37833333

Entwurf von Gate-Array-Schaltkreisen *Teil 3*

Prof. Dr. Dietmar Müller, Jan Pauliuk
Technische Universität, Karl-Marx-Stadt, Sektion Informationstechnik

Entwurfsvorgaben und -restiktionen bei Gate-Array-Schaltkreisen

Die Spezifik des Entwurfes und des Einsatzes von Gate-Array-Schaltkreisen erfordert die Einhaltung bestimmter Vorgaben und Restriktionen, damit der im Silizium realisierte Schaltkreis

– mit dem ersten Entwurf funktioniert und
– vom Hersteller (struktur-) prüfbar

ist.
Nachfolgend werden die aus dem Prinzip von Gate-Array-Schaltkreisen (siehe z. B. /1/, /2/) resultierenden allgemeinen Restriktionen und die durch das LSSD-Prinzip verursachten Vorgaben und Restriktionen beschrieben.

Allgemeine Vorgaben und Restriktionen

Wie bereits ausgeführt, existieren gegenwärtig nur digitale Funktionselemente (Hard- und Softwaremacros). Es fehlen neben direkten analogen Komponenten (wie Operationsverstärker) insbesondere auch Elemente mit Triggerfunktionen für die Eingangsstufen. Dieses Fehlen analoger Komponenten ist eine Schwachstelle, da üblicherweise viele der zu verarbeitenden Signale analogen Charakter haben.
Aus diesem Grunde werden künftige Gate-Array-Schaltkreise auch analoge Funktionselemente für Eingangs- und Ausgangsstufen ebenso aufweisen müssen wie RAM- und Prozessorstrukturen, damit die Einsatzbreite weiter erhöht werden kann.
Eine weitere Randbedingung ist die durch den Masteraufbau vorgegebene Anzahl der potentiellen Funktionselemente, die im allgemeinen noch in mögliche kombinatorische Funktionselemente und in mögliche LSSD-Flipflops (siehe nächsten Abschnitt) unterteilt werden, wodurch ebenfalls Entwurfsbeschränkungen entstehen. Der Anwender muß beim Entwurf seines Systemkonzepts weiterhin auf die verfügbaren Ein-/Ausgangsanschlüsse des Schaltkreises, die wesentlich durch die Gehäuseform bestimmt sind, achten. Für Gate-Array-Schaltkreise werden Chip-Carrier-Gehäuse mit bis zu 128 Pins verwendet (z. B. im System U 5200 PCC 64 oder künftig ausschließlich QFP 68). Bei weiter ansteigender Komplexität der Gate-Array-Schaltkreise wächst die Tendenz, auch Pin-Grid-Gehäuse mit bis zu 255 Pins einzusetzen, wobei die Gehäusekosten den Schaltkreispreis maßgeblich beeinflussen.
Die Ein-/Ausgangsanschlüsse müssen vom Anwender ebenfalls bezüglich ihrer konkreten Funktion „programmiert“ werden. Das heißt, der Anwender legt – wie bei den Funktionselementen – auch für die E-/A-Anschlüsse mit Hilfe der Interface-Macros fest, ob die um die Bondinsel angeordneten Schaltungen als Eingangsschaltung und/oder als Ausgangsschaltung fungieren sollen; dabei wird auch fixiert, mit welchem Pegel sie arbeiten, welches Treiberverhalten sie besitzen sollen u. a.

Restriktionen durch den prüfgerechten Entwurf

Da der Hersteller von Gate-Array-Schaltkreisen deren Funktion nicht kennt, das “Funktionieren“ des Schaltkreises – sprich das des Siliziums – dem Anwender aber nachweisen muß, ist die Notwendigkeit einer Strukturprüfung mit der Möglichkeit einer automatischen Testsatzgenerierung einleuchtend. Die Sinnfälligkeit einer Strukturprüfung wird durch die kleinen bis mittleren Stückzahlen pro Typ weiter untermauert.
Daher müssen Gate-Array-Schaltkreise so entworfen werden, daß sie trotz der unvermeidbaren Parameterschwankungen funktionieren und mit vertretbarem Aufwand prüfbar sind.
Die technologiebedingten Parameterschwankungen verändern insbesondere die dynamischen Kennwerte. Die zu entwerfenden Schaltungsstrukturen müssen jedoch unabhängig von diesen Schwankungen funktionieren, das heißt unabhängig von dynamischen Parametern sein, wodurch die Forderung nach pegelabhängigen (englisch: level sensitiv) Schaltungen entsteht. Die Prüfbarkeit des Schaltkreises wird durch die Ausbildung eines Scan-Pfades (englisch: scan-path) realisiert. Aus diesem Grunde werden Gate-Array-Schaltkreise im System U 5200 nach dem LSSD-Prinzip (englisch: level sensitive-scan-design) entworfen.

1. LS-Prinzip
Schaltungen sind nach /2/ dann level-sensitiv, wenn
– der Ausgangszustand (logischer Pegel) nach jeder erlaubten Eingangsänderung unabhängig von den internen Schaltungsverzögerungen ist
– sich der Ausgangszustand als statischer Endwert unabhängig von der Reihenfolge der Eingangsänderungen ergibt.
Daraus folgt für Gate-Array-Schaltkreise:
– ausschließliche Nutzung rückführungsfreier kombinatorischer Schaltungen, da diese stets level-sensitiv sind
– ausschließliche Nutzung synchron getakteter Speicherelemente, die durch den statischen Endwert (logischer Pegel) der kombinatorischen Ausgänge gesteuert werden, wodurch die innerhalb der Kombinatorik möglichen Hazards wirkungslos werden. Wettlauferscheinungen innerhalb dieser Speicherelemente müssen durch strukturelle Maßnahmen wirkungslos werden.
Mit derartigen synchron getakteten Speicherelementen und der rückführungsfreien Kombinatorik besitzt die Gesamtschaltung die geforderte Eigenschaft, *level sensitiv* zu sein.

2. Scan-Prinzip
Die Anwendung dieses Prinzips führt zu der für die Testung erforderlichen Steuer- und Beobachtbarkeit.
Steuerbarkeit: Das System kann in vorbestimmte Zustände überführt werden, wobei die Eingangs- und Zustandsvariablen einstellbar sein müssen.
Beobachtbarkeit: Die Reaktion des Systems (Ausgänge und Folgezustände) auf vorbestimmte Eingangssignale muß meßbar sein.
Dazu wird ein Scan-Pfad durch eine zeitweise Kettenschaltung aller obiger Speicherelemente realisiert. Dadurch wird die aufwendige Testung von sequentiellen Schaltungen in die weniger aufwendige Testung der Speicherelemente und die Testung der rückführungsfreien kombinatorischen Schaltungen getrennt.

3. LSSD-Schaltungsprinzip
Wie bekannt, sind die vorbereiteten Transistorstrukturen für die potentiellen Funktionselemente in parallelen Zeilen oder Straßen angeordnet (siehe zum Beispiel /3/).
Für die geforderten level-sensitiven Gesamtschaltungen, die aus der Einbettung rückführungsfreier Kombinatorik in getaktete FFs besteht, wird diese Anordnung genutzt. Dazu wird die Kombinatorik in bestimmten Zeilen konzentriert, und in den übrigen Zeilen sind die Speicherelemente angeordnet. Damit ergibt sich die im Bild 8 dargestellte LSSD-Schaltungsstruktur.

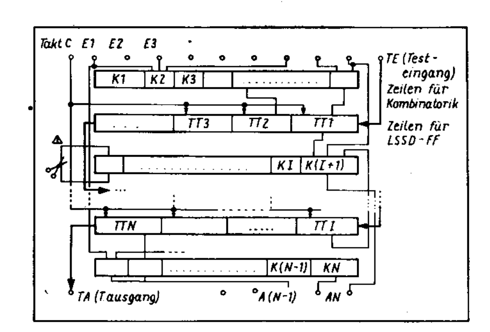


Bild 8 LSSD-Schaltungsstruktur

Dabei werden entweder einheitliche Transistoren auf dem gesamten Chip genutzt und durch die Verdrahtungsebenen zu kombinatorischen Funktionselementen oder zu Speicherelementen verbunden. Oder es werden speziell für die Kombinatorik und für die Speicherelemente entworfene und angepaßte Transistorstrukturen in Zeilen auf dem Chip erzeugt, wodurch die Lage und die Zahl der „Kombinatorikzeilen" und der „FF-Zeilen" fixiert sind (z. B. im System U 5200), jedoch wegen des speziellen Entwurfs – besonders bei den FFs – Chipfläche gespart oder die Schaltzeiten verkürzt werden.
Für diese LSSD-Schaltungsstrukturen muß gelten bzw. muß realisierbar sein:
– Eingänge der Kombinatorik werden von den Ausgängen der FFs oder von den Eingangsstufen des Schaltkreises gespeist
– Ausgänge der Kombinatorik werden an die Eingänge der FFs oder an die Ausgangsstufen des Schaltkreises geschaltet
– Rückführungen sind nur über die FFs zulässig
– Verbindung aller FFs zu einer Schiebekette während des Testmodus.
Die im Bild 8 gezeichneten LSSD-FFs haben eine Doppelfunktion. Im Arbeitsmodus (Systemmodus) sind sie „normale" getaktete Flipflops, und im Testmodus sind sie Flipflops einer Schiebekette. Dazu sind diese Master-Slave-FFs mit einem vor- und nachgeschalteten Umschalter für diese Betriebsarten ausgestattet.
Bild 9 zeigt das im Gate-Array-Lehrsystem /1/ genutzte aufwandsarme D-Master-Slave-FF, wobei das für die CMOS-Technologie charakteristische Transfergate genutzt wird. Mit dem Takt C wird bei C = 1 das Eingangssignal an D in den Master übernommen und bei C = 0 vom Master an den Slave weitergegeben.

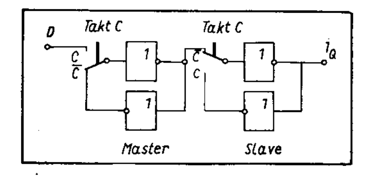


Bild 9 CMOS-typisches D-Master-Slave-FF

Durch den Einsatz weiterer Transfergates als Betriebsartenumschalter entsteht ein LSSD-FF (Bild 10).

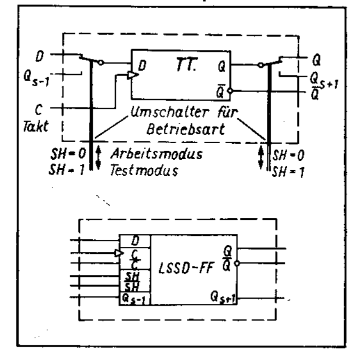


Bild 10 Schaltung und Symbol eines LSSD-FFs

Im Gate-Array-System U 5200 werden taktzustandsgesteuerte JK-Master-Slave-FFs innerhalb des LSSD-FF genutzt, dessen prinzipieller Aufbau im Bild 11 dargestellt ist. Es ist zu erkennen, daß im Gegensatz zu Bild 10 vor dem Slave-Teil umgeschaltet wird, woraus ein Arbeits- und ein Schiebeslave folgen. Die logische Schaltung dieses LSSD-FF ist beispielsweise in /4/ angegeben.

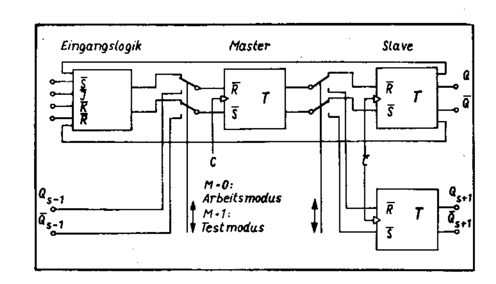


Bild 11 Prinzip des LSSD-FFs im System U 5200

Betriebsart: *Arbeitsmodus* (Anwendermodus)
Dieser Modus (SH = 0 im Bild 10 oder M = 0 im Bild 11) ist der normale, das heißt, die entworfene Schaltungsstruktur realisiert die zu erfüllende Funktion im Sinne eines synchron getakteten Systems. Die LSSD-FFs können und werden ausschließlich zur Erfüllung der sequentiellen Funktionen des Gate-Array-Schaltkreises genutzt.
Entsprechend Bild 10 wird während C = 1 der statische Endwert der vorgeschalteten Kombinatorik in den Master übernommen (beim U 5200 während der Masterphase wegen der Low-Aktivität bei C = 0).
Dabei sollte die Dauer dieser Taktphase nicht wesentlich über der minimalen, für eine sichere Zustandsübernahme ausreichenden Dauer liegen, um die maximal erreichbare Betriebsfrequenz nicht zu verringern und um die Wirkung zufälliger Spikes in der Kombinatorik zu begrenzen. Es sollte daher stets ein unsymmetrischer Takt mit der minimalen Masterphase angestrebt werden.
Während C = 0 wird der Zustand des Masters in den Slave übernommen und steht damit als Eingangssignal für die nachgeschaltete Kombinatorik zur Verfügung. Die Dauer dieser Taktphase (Slavephase) muß so gestaltet werden, daß neben der sicheren Zustandsübernahme in den Slave sich *auch* der statische Endwert an der nachgeschalteten Kombinatorik einstellen kann.
Neben den Übernahmezeiten in den Master und den Slave (einschließlich der Takttoleranzen und -verschiebungen) bestimmt die größte Verzögerungszeit der Kombinatorik und damit die maximale Kettenlänge der – vom Anwender entworfenen – Kombinatorik die maximale Betriebsfrequenz des Gate-Array-Schaltkreises.
Der Simulator des CAD-Systems (siehe zum Beispiel /5/) berechnet unter Worst-case-Bedingungen die Verzögerungszeiten der Kombinatorik, addiert die Zeiten für die Verzögerungen innerhalb des FFs und ermittelt damit die Dauer der Slavephase.

Betriebsart: *Testmodus* (Schiebemodus)
Durch das Signal SH = 1 (U 5200: M = 1) werden alle LSSD-Flipflops zu einer Schiebekette zusammengeschaltet.
Der damit mögliche (Struktur-)Test beim Hersteller beinhaltet:
– Testung der Speicherelemente
– Testung der (rückführungsfreien) Kombinatorik.
Erstere werden getestet, indem an den Test-

eingang QSI des Schaltkreises und damit an den Eingang QS des ersten, zu einer Kette geschalteten LSSD-FFs regelmäßige Testfolgen angelegt und durchgeschoben werden. Funktionieren alle FFs, erscheinen diese Folgen am Testausgang (bei U 5200 nach 102 Takten) unverfälscht.
Für die Testung der Kombinatorik werden vom CAD-System (siehe Teil 2) mittels eines modifizierten D-Algorithmus Testvektoren (-folgen) berechnet, die strukturelle Fehler – wie defekte Transistoren, Kurzschlüsse und Unterbrechungen von Verbindungsleitungen – prüfen.
Diese Folgen werden in die LSSD-Kette eingeschoben und Teile dieser Folgen direkt an die nutzerspezifischen Eingangspins des Schaltkreises gelegt.
Nach Umschalten auf den Arbeitsmodus stehen die Bitmuster an den Eingängen der Kombinatorik zur Verfügung (Steuerbarkeit!). Nach Erreichen des statischen Endwertes der Ausgangssignale der Kombinatorik werden diese in den Master und anschließend an den Slave übernommen. (Im Bild 11 ist der Modusumschalter nach dem Master nur zur Prinziperläuterung eingefügt. Schaltungstechnisch ist der Schiebeslave an den Master angeschlossen.)
Nach Umschalten in den Testmodus werden diese Ausgangszustände als Ergebnisse der kombinatorischen Verknüpfung der zu prüfenden IST-Struktur herausgeschoben (Beobachtbarkeit!) und können mit den berechneten SOLL-Ergebnissen verglichen werden. Bei Nichtübereinstimmung wird der Chip als fehlerhaft gekennzeichnet, sonst wird der obige Vorgang mit weiteren berechneten Testvektoren bis zur vollständigen Testung der Struktur fortgesetzt. (Bei redundanten Schaltungen läßt sich keine vollständige Testung erreichen, da keine Steuerbarkeit mehr vorhanden ist.)

Bild 12
Realisierung verschiedener FF-Arten

FF-TYP	Systemtakt	interner Untertakt
SR-FF	FF SR JK; SPG, SPG, R̄, S̄	FF SR JK; DNANDZ; R, S; SPG, SPG; TAKT
JK-FF	FF SR JK; J̄, K̄, SPG, SPG	DNANDZ; FF SR JK; J, K; SPG, SPG; TAKT
D-FF	INV1; FF SR JK; D; SPG, SPG	DNAND2; FF SR JK; D; SPG, SPG; TAKT
DV-FF	DNAND2; FF SR JK; D; V; SPG, SPG	NAND3; FF SR JK; NAND3; D; V; SPG, SPG; TAKT
T-FF	FF SR JK; T̄; SPG, SPG	DNAND2; FF SR JK; T; SPG, SPG; TAKT
Binärteiler	FF SR JK; ERDE; SPG, SPG	INV1; FF SR JK; TAKT; SPG, SPG

SPG ... Betriebsspannung ERDE ... Masse

Auswirkungen auf die Schaltungstechnik

Der Anwender muß beim Entwurf neben Kenntnissen aus den Datenblättern (z. B. bei U 5200: 1020 Gatteräquivalente, 102 RS-JK-MS-FFs, Stromverbrauch kleiner 10 mA, erreichbare Betriebsfrequenzen ...1 ...3 ...MHz) auch solche besitzen, die aus dem LSSD-Prinzip folgen.
Wesentliche Auswirkungen auf die Schaltungstechnik gegenwärtiger Gate-Array-Schaltkreise sind:
– Gate-Array-Schaltkreise arbeiten stets synchron, alle 102 FFs im System U 5200 werden durch einen Takt getrieben. Der Aufbau von Untertaktsystemen ist nur durch die Verwendung zusätzlicher Gatter möglich (z. B. alle FF-Eingänge durch Zweier-NANDs getort, wobei auch diese FFs selbstverständlich nur vom Systemtakt gesteuert werden – siehe Bild 12).
– Ausschließliche Nutzung der LSSD-FFs (Bilder 9 und 11) und von diesen abgeleiteten Varianten (im Sinne von verfügbaren Hardwaremakros) führt dazu, daß D-FFs und T-FFs nur mit zusätzlichem Aufwand erzeugt werden können. Bild 12 zeigt Varianten, die mit dem Systemtakt oder mit einem „Anwender"-Takt gesteuert werden.
– Verbot von Rückführungen in der Kombinatorik (Zyklenfreiheit) bedeutet Verbot des Aufbaus zusätzlicher Speicherelemente.
– Redundanz in den Schaltungen verhindert eine vollständige Prüfbarkeit. Der Anwender muß daher auf die Vermeidung von Redundanz achten, da bisher das CAD-System auch offensichtliche Redundanz (z. B. parallel geschaltete Eingänge) nicht signalisiert.
– Monoflops oder Verzögerungsschaltungen, die Gatterverzögerungszeiten nutzen, sind nicht zulässig und müssen durch Zähler oder Schieberegister realisiert werden.
Aus den verfügbaren Funktionselementen und der benutzten CMOS-Technologie ergeben sich:
• Komplexe Schaltungen müssen stets auf die Zusammenschaltung von in der Bibliothek vorhandenen Funktionselementen (Hard- und Softwaremakros) zurückgeführt werden, dabei ist
– Nutzung der negierenden und nichtnegierenden Ausgänge der Grundgatter und
– Nutzung von Mehrfach- und Komplexgattern anzustreben.
Mehrfachgatter nutzen die 8-Transistorzelle vollständig aus (z. B. QINV1 – vier Inverter; DNAND2, DNOR2 – zwei NAND- bzw. NOR-Gatter). Komplexgatter sind zweistufige UND-ODER- oder NAND-NOR-Strukturen, die ebenfalls die 8-Transistorzelle voll nutzen. Bei diesen Gattern werden die Chipflächen effektiv ausgenutzt (Verbesserung der Plazierung und Trassierung möglich) und die Verzögerungszeiten verkürzt (zellinterne Verdrahtung). Mehrfachgatter sollten insbesondere bei regulären, zusammengehörenden Schaltungsstrukturen angewendet werden, da bei ungünstigem Einsatz durch lange Verbindungen auch unnötige Verzögerungszeiten entstehen.
• E-/A-Stufen werden durch Interfacemakros realisiert, die CMOS- und/oder TTL-kompatible Pegel verarbeiten und die Twostate-, Tristate- oder Opendrain-Ausgänge besitzen. Es gibt bidirektionale Treiber zum Anschluß der Gate-Array-Schaltkreise an externe Bussysteme. Die internen Logikgatter besitzen kein solches Ausgangsverhalten, deshalb sind keine internen Bussysteme realisierbar.
• E-/A-Stufen sind nicht gelatcht. Bei Bedarf notwendiger Eingangs- und Ausgangsregister müssen diese aus den verfügbaren LSSD-FFs aufgebaut werden. Bei asynchronen Signalen an den Eingangspins sind zur Synchronisation zusätzliche Flankenauswerteschaltungen notwendig /6/.
• RAM- und ROM-Strukturen sind (noch) nicht verfügbar. Daher müssen diese bei Bedarf durch die verfügbaren LSSD-FFs oder über Multiplexkombinationen erzeugt werden, oder es müssen externe RAM-, ROM- oder EPROM-Schaltkreise genutzt werden.
• PLA-Strukturen (z. B. Dekoder) müssen in eine Randomlogik möglichst unter Nutzung von Komplexgattern umgewandelt werden.
• Hohe Ausgangslastfaktoren (größer 20) sind für die statische Funktion bei Anstieg der Verzögerungszeiten wegen der CMOS-Technologie möglich. Bei zu erfüllenden zeitlichen Vorgaben können zusätzliche Treiber eingefügt werden.

wird fortgesetzt

Literatur

/1/ Müller, D.: Gate-Array und Standardzelle – dominierende Vertreter innerhalb der ASICs. Mikroprozessortechnik, Berlin 3 (1989) 1, S. 21
/2/ Eichelberger, E. B.; Williams, T. W.: A logic design structure for LSI testability. Proc. 14th DAC. 1977, p. 462
/3/ Fischer, W.-J.; Gieseler, M.; Arndt, W.; Sorst, M.; Köhler, Th.: CMOS-Gate-Array-System U 5200. Nachrichtentechnik, Elektronik, Berlin 36 (1986) 1, S. 21
/4/ Benning, K.; Bürger, B.: ASICs 2. Radio, Ferns., Elektron., Berlin 38 (1989) 2, S. 93
/5/ Müller, D.; Fügert, E.: Entwurf von Gate-Array-Schaltkreisen (Teil 2). Mikroprozessortechnik, Berlin 3 (1989) 6, S. 168
/6/ Sorst, M.; Gieseler, M.; Fischer, W.-J.: Gate-Array-System U 5200. Mikroprozessortechnik, Berlin 1 (1987) 1, S. 4

KONTAKT

Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022, Tel. 5613195

Fraktale in Pascal

Berthold Biener, Erfurt

Im Beitrag „Fraktale vom KC 85/3" von Prof. H. Völz wurde in MP 1/88 eine interessante Einsatzmöglichkeit für grafikfähige Kleincomputer vorgestellt /1/. Die angeführten Rechenzeiten bis zu 250 Stunden für ein Bild erscheinen jedoch wenig akzeptabel. Deshalb wird bei vielen interessierten Nutzern der Wunsch aufgekommen sein, ein schnelleres Programm einzusetzen. Dem sind jedoch durch den Basic-Interpreter Grenzen gesetzt. Ein Ausweg ist die Verwendung einer anderen Programmiersprache auf Compilerbasis. Hier bietet sich die Sprache Pascal an, für die ein entsprechender Compiler auch für den KC 85/3 verfügbar ist. Damit lassen sich die Rechenzeiten auf weniger als 10% verkürzen. Das hier vorgestellte Programm benötigt für die Gesamtdarstellung des Apfelmännchens knapp 2 Stunden, während in Basic unter den gleichen Bedingungen 30 Stunden Rechenzeit verbraucht wurden.
Als neue Aspekte enthält das Programm eine weitere zur Fraktalberechnung verwendbare Funktion und eine neue Darstellungsform, beides basierend auf Arbeiten von Prof. Völz. Die Fraktale können nun auch dreidimensional dargestellt werden, wobei die dritte Dimension die Anzahl der erreichten Iterationen für jeden Punkt angibt. Das Programm wurde für KC-PASCAL 2.2 erstellt und am KC 85/3 erprobt (siehe Bild 7 auf Seite 237).
Die Prozedur PLOT übergibt Punktkoordinaten nach der Verknüpfung mit der Farbinformation (konstant gelb) an die PSET-Prozedur des Compilers. Die LINE-Prozedur zeichnet Linien mit der Einschränkung, daß bei betragsmäßigem Anstieg <1 der Verlauf von links nach rechts gehen muß. Diese kann jedoch zur allgemeinen Anwendung der Prozedur durch wenige zusätzliche Programmschritte aufgehoben werden.
Die Prozeduren APFEL und ROCHEN führen die eigentlichen Konvergenzuntersuchungen in einem Punkt aus. Dabei beinhaltet APFEL die bekannte Apfelmännchenfunktion, ROCHEN die angekündigte andere Funktion. Die Anzahl der Iterationen bis zum Abbruch wird über den Parameter M zurückgegeben.
Die Prozedur VELO realisiert die Anfangseinstellung des Minimum-Maximum-Vektors für die Unterdrückung verdeckter Linien in der dreidimensionalen Darstellung durch die Prozedur BERG. Letztgenanntes Unterprogramm wird für jeden zu berechnenden Punkt aufgerufen und prüft dessen Sichtbarkeit. Dabei wurden folgende Vereinfachungen vorgenommen: Die Verbindungslinie vom vorhergehenden Punkt zum aktuellen Punkt wird immer gezeichnet, wenn beide Punkte sichtbar sind, also auch wenn sie teilweise verdeckt wäre. Ist mindestens einer der beiden Punkte unsichtbar, wird keine Verbindungslinie gezeichnet, wodurch einzelne Peaks im Hintergrund als isolierte Punkte dargestellt werden. Auf Grund der hohen Liniendichte wirken sich diese Einschränkungen nicht nachteilig auf die Gesamtdarstellung aus.

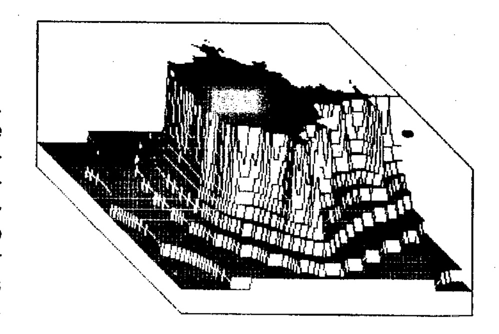


Bild 1 3-D-Darstellung des „Apfelmännchens"
Real: −1,20000.. 2,00000
Imag: −1,30000.. 1,30000
Für alle Bilder gilt: Divergenzgrenze 12
Maximale Iterationsanzahl 12

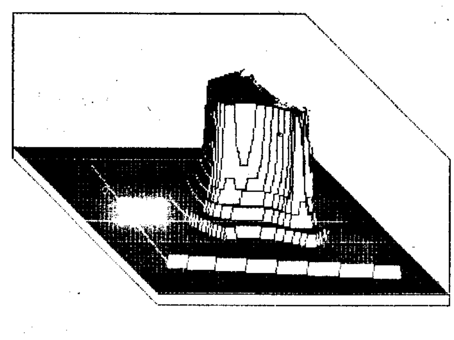


Bild 2 3-D-Darstellung des Konvergenzbereiches der „Rochen"-Funktion
Real: −3,00000..4,00000
Imag: −3,00000..4,00000

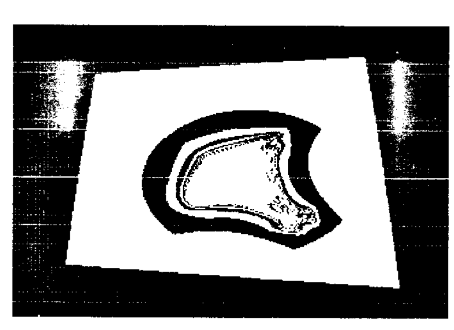


Bild 3 Plandarstellung unter der Verwendung der „Rochen"-Funkion
Real: −3,00000..4,00000
Imag: −3,00000..4,00000



Bild 4 Planardarstellung eines Teilbereiches der „Rochen"-Funktion
Real: 1,00000..2,00000
Imag: −1,00000..0,00000

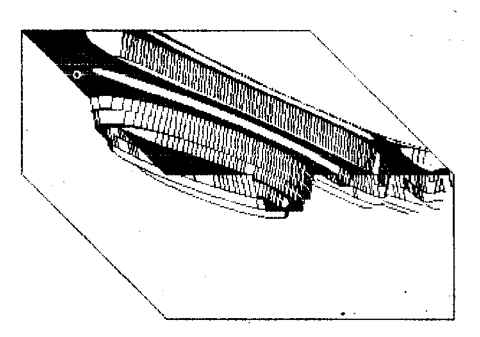


Bild 5 3-D-Darstellung eines Teilbereiches der „Rochen"-Funktion
(vergl. Bild 4 oben links)
Real: 1,10000..1,60000
Imag: −0,50000..−0,10000

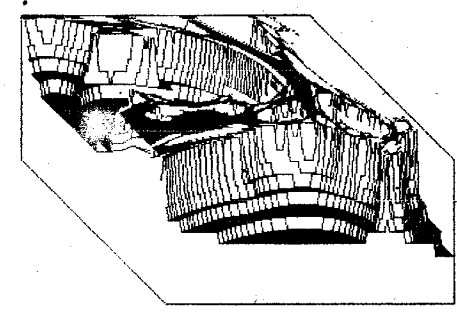


Bild 6 3-D-Darstellung eines Teilbereiches der „Rochen"-Funktion
(vergl. Bild 4 rechts)
Real: 1,60000..1,85000
Imag: −0,75000..−0,50000

Die Prozedur FELD bewirkt den Ablauf der Rechnung in horizontalen Linien. Die Zahl der Bildpunkte ist von der Darstellungsart abhängig. Für die dreidimensionale Darstellung wird mit 100 Linien zu 200 Punkten (Rechenzeit für Beispielgrafik: 10 min), für die planare Darstellung (Rechenzeit für Beispielgrafik: 30 min) mit 214 Linien zu 320 Punkten gearbeitet.
Das Hauptprogramm realisiert die Dateneingabe und einige Vorbereitungsarbeiten. Im Dialog erfolgen die Auswahl von Funktion und Darstellungsart, die Eingabe der Grenzen in X- und Y-Richtung, der maximalen Iterationsanzahl sowie der Divergenzgrenze. Das Programm benötigt neben 4 KBytes für den Quelltext nochmals ca. 4 KBytes für den Maschinencode. Es wurde speicherorientiert compiliert. Die Auskopplung eines selbständigen Maschinenprogramms gelang mit der verwendeten Compilerversion noch nicht.

Literatur

/1/ Völz, H: Fraktale vom KC 85/3, Mikroprozessortechnik, Berlin 2 (1988) 1, S. 27

/2/ Völz, H.: Basic für Fortgeschrittene Teil 1. Rundfunk der DDR, Berlin, Erstsendung: 9. 3. 1988

```
6716
6716   PROGRAM FRAKTALE;
6716   (*L-*)
6716   VAR
671F    G,H,I,J,K,L,M,N,ANZ,T,Z,S,Q,U,V,MD,TYP:INTEGER;
671F    X,Y,XM,XN,YM,YN,DX,DY,A,B,C,D,E,MAX:REAL;
671F    VEC:ARRAY[0..319,0..1] OF INTEGER;
671F    SB,SB0,SB1,SB2:BOOLEAN;
671F    F:CHAR;
671F   PROCEDURE PLOT(G,H:INTEGER);
6722   BEGIN
673A    PSET(G,H+1536)
6751   END;
675E   PROCEDURE LINE(G,H,I,J:INTEGER);
6761   VAR X,Y,DY:REAL;U,V:INTEGER;
6761   BEGIN
6779    IF H=J THEN
6791     BEGIN
6791      FOR U:=G TO I DO PLOT(U,H);
67DC     END;
67DC    IF G=I THEN
67F4     BEGIN
67F4      IF H>J THEN
680D       FOR U:=H DOWNTO J DO PLOT(G,U)
684A      ELSE FOR U:=H TO J DO PLOT(G,U);
68A5     END;
68A5    IF (G<>I) AND (H<>J) THEN
68D4     BEGIN
68D4      DY:=(0.0+J-H)/(0.0+I-G);
6931      IF ABS(DY)<1 THEN
6956       FOR U:=G TO I DO
6983        BEGIN
6986         V:=H+ENTIER(DY*(U-G));
69C9         PLOT(U,V);
69E0        END
69E0      ELSE
69E7       IF J>H THEN
6A00        FOR V:=H TO J DO
6A2D         BEGIN
6A30          U:=G+ENTIER((V-H)/DY);
6A73          PLOT(U,V);
6A8A         END
6A8A        ELSE FOR V:=H DOWNTO J DO
6ABD         BEGIN
6AC0          U:=G+ENTIER((V-H)/DY);
6B03          PLOT(U,V);
6B1A         END;
6B1E     END;
6B1E   END;
6B2D   PROCEDURE APFEL(X,Y:REAL;VAR M:INTEGER);
6B30   BEGIN
6B48    M:=0;
6B57    REPEAT
6B57     C:=A*A-B*B-X;D:=A*B;D:=D+D-Y;
6BEE     M:=M+1;A:=C;B:=D;
6C21    UNTIL (M>ANZ) OR (ABS(A)+ABS(B)>MAX);
6C71   END;
6C7B   PROCEDURE ROCHEN(X,Y:REAL;VAR M:INTEGER);
6C7E   VAR U,V:REAL;
6C7E   BEGIN
6C96    M:=0;
6CA5    REPEAT
6CA5     U:=A*A;V:=B*B;
6CE6     C:=U-V-Y;D:=U+V-X;
6D5C     M:=M+1;A:=C;B:=D;
6D8F    UNTIL (M>ANZ) OR (ABS(A)+ABS(B)>MAX);
6DDF   END;
6DEE   PROCEDURE VELO;
6DF1   BEGIN
6E09    FOR I:=0 TO 319 DO
6E23     BEGIN
6E26      VEC[I,0]:=0;VEC[I,1]:=255;
6E98     END;
6E9B   END;
6EA1   PROCEDURE BERG(X,Y,M,ZMAX:INTEGER);
6EA4   VAR U,V:INTEGER;
6EA4   BEGIN
6EBC    IF X=0 THEN SB0:=FALSE;
6ED5    U:=110+X-Y;
6EFB    V:=10+Y+ENTIER(100.0*(M-1)/ZMAX);
6F4E    SB1:=V<=VEC[U,1];
6F99    SB2:=V>=VEC[U,0];
6FE5    SB:=SB1 OR SB2;
6FF1    IF SB1 THEN VEC[U,1]:=V;
7037    IF SB2 THEN VEC[U,0]:=V;
707D    IF SB AND NOT SB0 THEN PLOT(U,V);
70A3    IF SB AND SB0 THEN LINE(J,G,U,V);
70CF    IF SB THEN
70D6     BEGIN J:=U;G:=V END;
70E8    SB0:=SB;
70EE   END;
70FA   PROCEDURE FELD(Z,S,MD,TYP:INTEGER);
70FD   BEGIN
7115    K:=0;Y:=YM-DY;
7139    WHILE (K<Z) AND (INCH<>'X') DO
7162     BEGIN
7162      Y:=Y+DY;X:=XM-DX;L:=0;
71A0      WHILE L<S DO
71BA       BEGIN
71BA        X:=X+DX;A:=0.5;B:=0.0;
71EE   IF TYP=1 THEN APFEL(X,Y,Q) ELSE ROCHEN(X,Y,Q);
7244        IF MD=2 THEN BERG(L,K,Q,ANZ)
7269        ELSE BEGIN IF (Q MOD 2)=0 THEN PLOT(L,K+8) END;
72A8        L:=L+1;
72AF       END;
72B2      K:=K+1;
72B9     END;
72BC   END;
72C6   BEGIN
72CF   REPEAT
72CF    WRITELN(CHR(12),'FRAKTALE in PASCAL auf KC 85/3');
7305    FOR I:=1 TO 40 DO WRITE('=');
732A    WRITELN('(C) bidat 1988');
7346    WRITELN;WRITELN('FUNKTIONEN:');
7362    WRITELN(' 1 = APFELMAENNCHEN');
7383    WRITELN(' 2 = ROCHEN');
739C    WRITELN(' 0 = ENDE');
73B3    WRITE('AUSWAHL ? ');READ(TYP);
73CE   IF TYP>0 THEN
73E1   BEGIN
73E1    WRITELN;WRITELN('DARSTELLUNG:');
73FE    WRITELN(' 1 = PLANAR');
7417    WRITELN(' 2 = GEBIRGE');
7431    WRITE('AUSWAHL ? ');READ(MD);
744C    WRITELN;WRITELN('PARAMETER:');
7467    WRITE('REALTEIL MIN:');READ(XM);
7489    WRITE('REALTEIL MAX:');READ(XN);
74AB    WRITE('IMAGTEIL MIN:');READ(YM);
74CD    WRITE('IMAGTEIL MAX:');READ(YN);
74EF    WRITE('ITERATIONSANZ:');READ(ANZ);
750E    WRITE('ABBRUCHGRENZE:');READ(MAX);
7531    Z:=214;S:=320;
753D    WRITELN(CHR(12),'FRAKTALE KC 85/3  (C) bidat 1988');
7572    IF MD=2 THEN
7584     BEGIN
7584      Z:=100;S:=200;VELO;
7595      LINE(10,210,10,110);
75AA      LINE(10,110,110,10);
75BF      LINE(110,10,310,10);
75D4      LINE(310,10,310,110);
75E9      LINE(310,110,210,210);
75FE      LINE(10,210,210,210)
760E     END;
7613    DX:=(XN-XM)/S;
763C    DY:=(YN-YM)/Z;
7665    FELD(Z,S,MD,TYP);
767A   WRITELN('REAL:',XM,'..',XN);
76AE   WRITELN('IMAG:',YM,'..',YN);
76E2   READ(F);
76E8    END;
76E8   UNTIL TYP=0;
76FA   END.
```



TERMINE

Fachtagung „Probleme der Organisation beim Aufbau von CIM-Betrieben"
WER? Bezirksverband der Kammer der Technik Halle
WANN? 1. November 1989
WO? Halle
WAS?
- Vorstellung von Ansätzen für CIM-Betriebe, Konzeptionen, Stand
- Beispiele aus der Praxis

WIE? Anfragen richten Sie bitte an: KDT, BVO Halle, Bereich W/T, Geschwister-Scholl-Straße 39, PF 119, Halle, 4060; Tel. 37136.

Große

Der Bezirksfachausschuß Mikroelektronik beim Bezirksvorstand Berlin der KDT führt im zweiten Halbjahr 1989 Spezialkonsultationen zu folgenden Themen durch:

20. Sept.	Mehrprozessorsysteme für Echtzeitverarbeitung
11. Okt.	PC-Einsatz für den Leiterplattenentwurf und für die mechanische Konstruktion
8. Nov.	Entwicklungsumgebung und Anwendungsbeispiele für den digitalen Signalprozessor
20. Dez.	Anwendungen von digitalen Signalprozessoren

Änderungen sind dem Bezirksfachausschuß Mikroelektronik vorbehalten.
Die Teilnahme an den Konsultationen bedarf keiner Anmeldung und ist kostenlos.
Die Konsultationen finden jeweils 14.00 Uhr im Haus der KDT, 1080 Berlin, Kronenstraße 18, statt.

Driese

Fachtagung „Neue Erkenntnisse der Mikroelektronik/Mikrorechentechnik"
WER? Bezirksverband der Kammer der Technik Halle
WANN? 19. und 20. September 1989
Wo? Halle
WAS?
- Industriecomputer (16 und 32 Bit)
- Prozessortechnik, Betriebssysteme
- Grafiksysteme
- Softwaretechnologien

WIE? Anfragen richten Sie bitte an: KDT, BVO Halle, Bereich W/T, Geschwister-Scholl-Straße 39, PF 119, Halle, 4060; Tel. 37136

Große

Bildschirmsteuerung unter Turbo-Pascal 4.0 und DCP

Dr. Peter Zschockelt

Die standardmäßig für monochrome Bildschirme in Turbo-Pascal vorhandenen Ausgabemöglichkeiten beschränken sich auf eine unterschiedliche Helligkeit der Zeichen. Für optisch ansprechende Bildschirmdarstellungen ist jedoch auch die Nutzung von inversen, blinkenden und unterstrichenen Zeichen wünschenswert. Dazu sind eigene Ausgaberoutinen erforderlich. Nachfolgend wird eine mögliche, einfache Lösung vorgeschlagen, die auch für Anwender mit geringen Betriebssystemkenntnissen verständlich ist. Sie basiert auf der Nutzung der DOS-Funktion 09H (Übergabe von Zeichenketten an das Betriebssystem). Bewußt wurde die Prozedur **Writi** im nachfolgenden Programm-Quelltext ähnlich der standardmäßigen Pascal-Prozedur **Write** gestaltet. Die Ausgabe von Zeichen auf den Bildschirm kann gemischt mittels Write, Writeln oder Writi (bzw. im Beispiel mittels Textxy) erfolgen.

Voraussetzung für eine Nutzung ist aber, daß vom Betriebssystem die ANSI-Steuersequenzen (Datei ANSI.SYS) geladen wurden. Diese Voraussetzung dürfte jedoch bei der üblichen Betriebsweise der 16-Bit-Rechner unter DCP im allgemeinen erfüllt sein.

Der nachfolgende Programm-Quelltext wurde für die Version 4.0 von Turbo-Pascal geschrieben. In dieser Version ist als Interface zur Nutzung der Pascal-Prozeduren MsDos und Intr der Datentyp

```
Registers=RECORD
   case Integer of
   0: (AX, BX, CX, DX, BP, SI,
      DI, DS ES, Flags: Word);
   1: (Al, AH, BL, BH, CL, CH,
      DL, DH: Byte);
   END;
```

standardmäßig vorhanden. In der Version 3.xx muß hingegen die Vereinbarung des Records durch den Programmierer erfolgen. Auf weitere Unterschiede wird – soweit erforderlich – im Quelltext mit Kommentaren hingewiesen.

Im Interesse der leichten Verständlichkeit wurde in der dargestellten Lösung das Sonderfunktionen erfüllende Währungszeichen 24H (Abschluß der übergebenen Zeichenkette) einfach durch das Zeichen @ ersetzt. Für die Mehrzahl aller Anwendungen dürfte das ausreichen. Eine komfortable Lösung ist möglich, wenn in der Ausgabekette vorhandene Währungszeichen als Einzelzeichen mit der Betriebssystemfunktion 02H (an Stelle von 09H) ausgegeben werden. Dazu muß das Zeichen im Register DL stehen. Die von der Funktion 09H benutzten Register DS und DX werden hier nicht belegt.

Die dargestellte Lösung ist kompatibel zu allen Rechnertypen unter DCP. Sie ermöglicht ferner eine relativ einfache Umstellung der für den A 7100 entwickelten Turbo-Pascal-Programme auf DCP. Bei der Neuentwicklung von Programmen können unter der Voraussetzung einer homogenen Hardwarebasis natürlich auch die Turbo-Pascal-Prozeduren Textbackground und Textcolor benutzt werden.

```
{$I+}    {I/O-Pruefung an}

{ Demonstrationsprogramm zu den Prozeduren                    }
{ "Bildschirmausgabe mit Sonderfunktionen unter TURBO ´4      }
{ Hochschule fuer Oekonomie * Sektion Wirtschaftsinformatik }
{                           * Programmierer:     Zschockelt }

Program DEMO4;
uses                            { entfaellt fuer Version 3.xx }
  Crt,
  Dos;
TYPE
     Str_8  =STRING[8];
     BS_Typ =STRING[80];
     Str_121=STRING[121];
VAR
     ESC,Norm,Hell,Inv,NUnt,HUnt,NBli,HBli:Str_8;

{ ---------------------------------------------------------- }
{ SPEZIELLE PROZEDUREN ZUR BILDSCHIRMAUSGABE VON ZEICHEN IN }
{ INVERSER, BLINKENDER UND UNTERSTRICHENER DARSTELLUNG      }
{ ---------------------------------------------------------- }

{ Initialisierung der Konstanten zur Bildschirmsteuerung    }

PROCEDURE BildInit
          (Var ESC,Norm,Hell,Inv,NUnt,Hunt,NBli,HBli:Str_8);
CONST CNorm='[0m'; CHell='[1m'; CInv='[7m'; CUnt='[4m';
      CBli='[5m';
BEGIN
  ESC:=CHR(27); Norm:=ConCat(ESC,CNorm);
  Hell:=ConCat(ESC,CHell); Inv:=ConCat(Norm,ESC,CInv);
  NUnt:=ConCat(Norm,ESC,CUnt); HUnt:=ConCat(Hell,ESC,CUnt);
  NBli:=ConCat(Norm,ESC,CBli); HBli:=ConCat(Hell,ESC,CBli);
END;

{ Ausgabe mit ESCAPE-Folgen  ( DOS-Funktion 09H )          }

PROCEDURE WRITI (INFORMATION : Str_121);
VAR  ESC_sequenz:string[121];
     regs:Registers;
     { in Version 3.xx Register selbst definieren,       }
     { Typ Integer statt Word, z. B.:                    }
     { TYPE Registers=record ax,bx,...,es,flags:integer; }
     { oder als RECORD mit variantem Teil (free unions), }
     { analog zum Standard in TURBO PASCAL 4.0           }
     i:integer;
     KB:string[120];
BEGIN
KB:=INFORMATION;
{ Ersatz der Waehrungszeichen 24H (Pascal=#36/$24) durch @ }
i:=Pos(#36,KB);
While i>0 do begin
               Delete(KB,i,1); Insert('@',KB,i);
               i:=Pos(#36,KB)
             end;
{ Ende der Zeichenumwandlung #36 (24H) in  @ . Wenn        }
{ diese Loesung unbefriedigend ist, DOS-Funktion 02H mit }
{ Uebergabe des Zeichens #36 in Register dl verwenden    }

ESC_sequenz:=concat(KB,#36);
regs.ax:=$0900;    { '$' (Dollarzeichen) steht fuer das    }
                   { auf der jeweiligen Tastatur uebliche  }
                   { Pascal-Zeichen zur Bezeichnung hexa-  }
                   { dezimaler Ziffern                     }
regs.ds:=seg(ESC_sequenz);
regs.dx:=ofs(ESC_sequenz)+1;
MsDos(regs);                       { Ausgabe Zeichenkette }
END;

{ Bildschirmausgabe ab Position x,y                         }

PROCEDURE Textxy (x,y:integer; zeile:bs_typ);
 BEGIN Gotoxy(x,y); Writi(zeile); END;

{ --------------------------------------------------------- }
{ ENDE DER PROZEDUREN                                       }
{ Fortsetzung des Demonstrationsprogramms                   }
{ --------------------------------------------------------- }

BEGIN
ClrScr;
BildInit(ESC,Norm,Hell,Inv,NUnt,HUnt,NBli,HBli);
Textxy(2,1,Inv+' DEMONSTRATIONSPROGRAMM ZUR VERWENDUNG  +
       'DER BILDSCHIRMAUSGABE-PROZEDUREN '+Norm);
Textxy(2,5,Norm+' Normaler            Text '+Norm);
Textxy(2,6,Hell+' Heller              Text '+Norm);
Textxy(2,7,Inv +' Inverser            Text '+Norm);
Textxy(2,8,NUnt+' Normal-Unterstr.    Text '+Norm);
Textxy(2,9,NBli+' Normal-Blinkender Text '+Norm);
Textxy(25,11,Inv+' ENDE DER DEMONSTRATION '+Norm);
END.

{ --------------------------------------------------------- }
{ Ende des Demonstrationsprogramms                          }
{ --------------------------------------------------------- }
```

TERMINE

2. MIDI-Informationsbörse

WER? Kulturhaus im Ernst-Thälmann-Park, Computerclub
WANN? 29. Oktober 1989, 10.00 bis 18.00 Uhr
WO? Berlin

WAS? Rund um den MIDI-Standard wird informiert, vorgetragen und ausgetauscht.
WIE? Schriftliche Anmeldungen, Themenwünsche und Vorschläge für Beiträge richten Sie bitte unter dem Kennwort MIDI an: Kulturhaus im Ernst-Thälmann-Park, Computerclub, Dimitroffstraße 101, Berlin, 1055.

Schellhorn

Turbo-Pascal-Praxis

Teil 2

Manfred Zander, Dresden

Im Teil 1 unseres Kurses (s. MP 6/89, S. 175) wurden die Programmierung einer positionierten, absturzfesten Eingabe und die Auswahl mit Menüs dargestellt sowie die Arbeit mit Fenstern eingeleitet. Wir endeten mit dem Bild 9 und dem Vorstellen der Grundprozeduren. Der Teil 2 setzt nun hier fort und behandelt den Komplex Fenstertechnik.

Mit der Prozedur inslinedown (vgl. Bild 9 im Teil 1) ist es nun möglich, unter einem Tabellenkopf, der fest auf dem Bildschirm steht, immer wieder neue Zeilen zu erfassen, und bereits erfaßte auch unten wegrollen zu lassen. Hier ein kurzer Programmauszug, der dies zeigt.

```
writexy(1,10,'Name                Vorname');
writexy(1,11,'--------------------------------');
repeat
  inslinedown(12);
  read(namen[i]);gotoxy(20,12);
  read(Vornamen[i];i:=i+1
until namen [i–1]='';
```

Besteht der Wunsch, das Wegrollen der Zeilen nur bis zu einer bestimmten Zeile zuzulassen, das heißt sowohl einen Tabellenkopf als auch einen Tabellenfuß fest auf dem Bildschirm zu belassen, kann die Prozedur inslinedownbis (Bild 9) genutzt werden. Sie schränkt den Bereich vertikal ein, in dem das Wegrollen der Zeilen erfolgen soll. Zuerst errechnet sie die Anzahl der zu verschiebenden Bildschirmzeichen, die in der Variablen z gespeichert wird. Für die Variablen u und w werden Offsetwerte bestimmt, gerechnet ab der Adresse des Bildschirmanfangs. Der Offsetwert u gibt dabei die Adresse für das erste zu verschiebende Zeichen, und der Offsetwert w die neue Addresse für dieses Zeichen an. Die Errechnung der wirklichen Adressen erfolgt erst in der inline-Anweisung, in der wieder ein zyklischer Ladebefehl des Prozessors genutzt wird.

Die beiden Parameter der Prozedur inslinedownbis spezifizieren in ihrer Reihenfolge die erste Zeile, die wegrollen soll, also die Position der einzufügenden Leerzeile und die letzte wegzurollende Zeile, das heißt die Zeile, die aus dem Bereich herausgerollt und damit vom Bildschirm gelöscht werden soll. Damit ist dies die erste Prozedur, die eine Art Fenster auf dem Bildschirm behandelt. Der Aufruf:

Inslinedownbis(10,13);

würde die folgende Zeilenbewegung hervorrufen:

10. Zeile auf die 11. Zeile
11. Zeile auf die 12. Zeile
12. Zeile auf die 13. Zeile

Die 10. Zeile ist danach leer, und der Inhalt der ursprünglichen 13. Zeile ist nicht mehr auf dem Bildschirm vorhanden. Wir haben also eine vertikale Fensterbegrenzung erreicht.

Nun gilt es auch noch, eine horizontale Eingrenzung zu erreichen, wie es ja für ein Fenster typisch ist. Als erstes wollen wir die Prozedur inslinedownin vorstellen (Bild 10). In ihr wird zuerst wieder die Länge der Teilzeilen bestimmt, die auf dem Bildschirm nach unten verschoben werden sollen. Da diese Teilzeilen nun nicht mehr lückenlos im Hauptspeicher hintereinander stehen, muß die Berechnung der Offsetadressen für jede Zeile, die verschoben werden soll, einzeln erfolgen. Mit der Inline-Anweisung wird nun auch jeweils nur eine Teilzeile auf ihre neue Position gebracht.

Das abschließende Löschen der einzufügenden Zeile am oberen Rand des zu behandelnden Fensters kann nun nicht mehr durch die Ausgabe eines Bildschirmsteuerzeichens erfolgen. Wir sind gezwungen, jedes einzelne Zeichen dieser Fensterzeile mit einem Leerzeichen zu überschreiben und den Kursor danach an ihren Anfang zu positionieren.

Die Prozedur inslinedownin benötigt nun bereits vier Parameter, die den zu verschiebenden Bereich eingrenzen. X1 und Y1 definieren den oberen linken Eckpunkt des Bereiches, die Parameter X2 und Y2 den unteren Eckpunkt des zu rollenden Bereiches auf dem Bildschirm. Mit dieser Prozedur sind wir jetzt in der Lage, die oberste Zeile des angegebenen Fensters mit einer Leerzeile zu füllen und alle anderen im Fenster enthaltenen Zeilen nach unten wegrollen zu lassen. Dabei bleibt der komplette Bildschirm, der nicht innerhalb der beiden Eckpunkte liegt, unverändert. Ein typischer Aufruf der Prozedur wäre beispielsweise

Inslinedownin(30,10,70,15);

Durch geeignete Wahl der Eckpunkt-Parameter kann mit dieser Prozedur auch die Funktion der bereits vorgestellten Prozeduren inslinedown und inslinedownbis nachgestaltet werden. Natürlich wollen wir auch in der Lage sein, innerhalb eines Fensters auf der letzten Zeilenposition eine Leerzeile einzufügen und dabei alle anderen Zeilen innerhalb des Fensters nach oben wegrollen zu lassen. Dies wird mit der folgenden Prozedur inslineupin erreicht. Da ihre innere Struktur fast völlig der Prozedur inslinedownin entspricht, muß ihre Funktion nicht näher erläutert werden.

```
procedure inslinedownin(x1,y1,x2,y2:integer);
  var i,o,u,z:integer;
  begin
    z:=x2-x1+1;
    for i:=y2 downto y1+1 do begin
      u:=(i-1)*80+x2-1; o:=u-80;
      inline($21/$00/$f8/  (*LD HL,BS-Anfang   *)
             $ED/$4b/u/    (*LD BC,(u)         *)
             $09/          (*ADD HL,BC         *)
             $EB/          (*EX DE,HL          *)
             $21/$00/$f8/  (*LD HL,BS-Anfang   *)
             $ed/$4b/o/    (*LD BC,(o)         *)
             $09/          (*ADD HL,BC         *)
             $ed/$4b/z/    (*LD BC,(z)         *)
             $ed/$b8);end; (*LDDR              *)
    gotoxy(x1,y1);
    for i:=x1 to x2 do write(' ');
    gotoxy(x1,y1)
  end;


procedure inslineupin(x1,y1,x2,y2:integer);
  var i,o,u,z:integer;
  begin
    z:=x2-x1+1;
    for i:=y1+1 to y2 do begin
      u:=(i-1)*80+x2-1; o:=u-80;
      inline($21/$00/$f8/   (*LD HL,BS-Anfang   *)
             $ED/$4b/o/     (*LD BC,(o)         *)
             $09/           (*ADD HL,BC         *)
             $EB/           (*EX DE,HL          *)
             $21/$00/$f8/   (*LD HL,BS-Anfang   *)
             $ed/$4b/u/     (*LD BC,(u)         *)
             $09/           (*ADD HL,BC         *)
             $ed/$4b/z/     (*LD BC,(z)         *)
             $ed/$b8);end;  (*LDDR              *)
    gotoxy(x1,y2);
    for i:=x1 to x2 do write(' ');
    gotoxy(x1,y2)
  end;
```

Bild 10 Grundprozeduren der Fenstertechnik

Arbeiten mit den Fenstern

Mit den beiden Prozeduren inslinedownin und inslineupin haben wir jetzt die notwendigen Hilfsmittel, um auf dem Bildschirm mit Fenstern operieren zu können. Die Benutzung dieser Prozeduren für mehere Fenster ist aber recht mühsam, da ihnen stets die Eckpunktkoordinaten des zu behandelnden Fensters übergeben werden müssen. Es ergibt sich also die Notwendigkeit, auch einige logische Hilfsmittel zur Nutzung der Fenster zu entwerfen, die uns die Verwaltung der konkreten Koordinaten abnehmen. Dies wird vor allem dann notwendig, wenn wir die Fenster samt Inhalt über den Bildschirm verschieben wollen, da sich die aktuellen Eckpunkte nach jeder Verschiebeoperation natürlich ändern. Zur Verwaltung der Fensterkoordinaten nutzen wir ein Feld, in dem die Informationen gesammelt werden, die die jeweils deklarierten Fenster kennzeichnen (Bild 11).

In dem definierten Array wins können nun die Daten von fünf Fenstern gespeichert werden. Dabei ist jedes Fenster durch vier Integerwerte definiert, die die Eckpunkt-Koordinaten der Fenster repräsentierten, sowie durch ein Zeichen, in dem wir den ASCII-Wert speichern wollen, mit dem das Fenster bei Bedarf umrahmt werden soll. Zum Füllen dieser Felder nutzen wir die folgende Prozedur windowdef. Durch diese Vereinbarungen zur Verwaltung der Fensterwerte können wir nun die Bedienung der Prozeduren inslinedownin und inslineupin wesentlich vereinfachen. Die beiden Prozeduren InWindowDown und InWindowUp übernehmen die korrekte Füllung der Parameterlisten der Prozeduren inslinedownin und inslineupin mit den Koordinaten, die für das zu behandelnde Fenster im Array wins gespeichert sind.

Wurden mit Hilfe der Prozedur windowdef die Koordinaten der Fenster einmalig festgelegt, braucht sich der Programmierer also im weiteren nicht mehr um die konkreten Koordinaten der einzelnen Fenster zu kümmern. Durch die Anwendung der neuen Prozeduren InWindowDown und InWindowUp genügt es völlig, die Nummer des zu behandelnden Fensters anzugeben. Diesen Service wollen wir auch für alle weiteren Fenster-Behandlungen erreichen. Als nächstes wollen wir




```
const FensterAnzahl = 5;
var   wins : array[1..FensterAnzahl] of
             record
               x1,y1,x2,y2:integer;
               c:char
             end;


procedure windowdef(i,x1,y1,x2,y2:integer;c:char);
  begin
    wins[i].x1:=x1;      wins[i].x2:=x2;
    wins[i].y1:=y1;      wins[i].y2:=y2;
    wins[i].c:=c
  end;


procedure InWindowDown(i:integer);
  begin
    inslinedownin(wins[i].x1,wins[i].y1,wins[i].x2,wins[i].y2)
  end;


procedure InWindowUp(i:integer);
  begin
    inslineupin(wins[i].x1,wins[i].y1,wins[i].x2,wins[i].y2)
  end;
```

Bild 11 Logische Bedienung der Fenster

```
procedure Rahmen(x1,y1,x2,y2:integer;c:char);
  var i:integer;
  begin
    write(kursoroff);gotoxy(x1,y1);
    for i:=x1 to x2 do if c='|' then write('-') else write(c);
    for i:=y1+1 to y2-1 do
      begin
        gotoxy(x1,i);write(c); gotoxy(x2,i);write(c)
      end;
    gotoxy(x1,y2);
    for i:=x1 to x2 do if c='|' then write('-') else write(c);
    write(kursoron)
  end;


procedure WindowWrite(i:integer);
  begin
    Rahmen(wins[i].x1-1,wins[i].y1-1,wins[i].x2+1,wins[i].y2+1,wins[i].c)
  end;
```

Bild 12 Umrahmen von Fenstern

```
procedure WriteDown(i:integer;str:string80);
  var l:integer;
  begin
    InWindowDown(i);
    l:=wins[i].x2-wins[i].x1+1;
    if length(str)> l then str:=copy(str,1,l);
    write(str)
  end;


procedure WriteUp(i:integer;str:string80);
  var l:integer;
  begin
    InWindowUp(i);
    l:=wins[i].x2-wins[i].x1+1;
    if length(str)> l then str:=copy(str,1,l);
    write(str)
  end;


function holeRin(i:integer;str:string80):real;
  var Io:integer;
      s:string80;
      r:real;
  begin
    repeat
      writeUp(i,str+'=');        (* Ausgabe der Eingabeaufforderung *)
      read(s);                   (* Einlesen eines Strings          *)
      val(s,r,Io)                (* Umwandlung des Strings in real  *)
    until Io=0;                  (* bis Umwandlung Io ist           *)
    holeRin:=r
  end;


procedure ReadWriteInWindow(i:integer;var str:string80);
  var l,d:integer; c:char;
  begin
    inwindowdown(i); str:='';d:=0;
    l:=wins[i].x2-wins[i].x1+1;
    repeat
      read(kbd,c);
      if c<>#13 then write(c);
      str:=str+c;d:=d+1;
      if c = #8 then
        begin
          str:=copy(str,1,length(str)-2);
          write(#$20,#8);d:=d-2;
        end;
      if c = #127 then d:=d-2
    until (c = #13) or (l=d)
  end;
```

Bild 14 Schreiben und Lesen in Fenstern

unsere Fenster umrahmen. Dazu schreiben wir zuerst die Prozedur Rahmen (Bild 12), die die konkreten Koordinaten benötigt, und dann die Prozedur WindowWrite, die die erste leichter bedienbar macht. Die Prozedur Rahmen realisiert das Umrahmen eines Bildschirmbereiches mit dem angegebenen ASCII-Zeichen. Eine Sonderbehandlung erfährt das Zeichen |. Wird es angegeben, so werden die waagerechten Begrenzungen nicht mit dem ASCII-Zeichen selbst, sondern mit einem Minus-Zeichen aufgebaut. Die Prozedur WindowWrite realisiert in bewährter Weise die Bereitstellung der konkreten Koordinaten und des ASCII-Zeichens, welches für den Rahmen verwendet werden soll. Zu ihrer Nutzung ist wiederum lediglich die Angabe der Fenster-Nummer notwendig. Hier nun wollen wir ein kurzes Beispiel für die Nutzung und die Wirkung der bisherigen Prozeduren angeben:

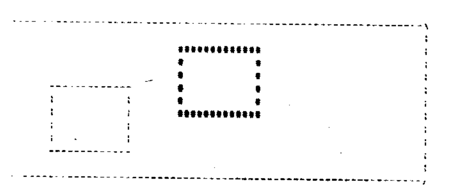


Bild 13 Bildschirm mit umrahmten Fenstern

```
...
Windowdef(1,10,3,20,6,'|');
Windowdef(2,30,6,40,9,'#');
WindowWrite(1);
WindowWrite(2);
...
```

Mit Hilfe dieser vier Anweisungen wird das in Bild 13 veranschaulichte Bildschirmaussehen realisiert.

In Bild 14 folgen zwei Prozeduren, mit denen wir in diese nun definierten und auf dem Bildschirm gekennzeichneten Fenster schreiben können. Es sind deshalb zwei Prozeduren vorgesehen, da wir ja das Rollen in den Fenstern sowohl aufwärts als auch abwärts ermöglichen wollen. Die Prozedur WriteDown, der die Nummer des betreffenden Fensters und der auszugebende Text übergeben wird, rollt zuerst den kompletten Fensterinhalt um eine Zeile nach unten. Der neu in das Fenster zu schreibende Text wird dann in die geschaffene Leerzeile am oberen Rand des Fensters geschrieben. Die Prozedur WriteUp dagegen schiebt den Fensterinhalt nach oben und schreibt auf der untersten Fensterzeile.

In beiden Prozeduren wird garantiert, daß zu langer Text nicht über das jeweilige Fenster hinaus geschrieben wird, sie sorgen selbständig für eine neue Zeile im Fenster, auf der der Text dann geschrieben wird. Bild 15 zeigt das erzeugte Bildschirmaussehen. Selbstverständlich müssen die genutzten Fenster 1 und 2 vor diesen Anweisungen erst mit der Prozedur windowdef deklariert und mit der Prozedur WindowWrite gezeichnet worden sein.

```
...
writeDown(1,'Erster');
writeUp(2,'Zweiter');
writeDown(1,'Dritter');
writeUp(2,'Vierter');
writeUp(2,'Zu langer Text für das Fenster');
...
```

So wie wir mit den Prozeduren WriteDown und WriteUp in die Fenster geschrieben haben, ist es natürlich auch möglich, in den Fenstern positioniert zu lesen. Allerdings ist es hierbei kaum möglich, Prozeduren zu entwik-

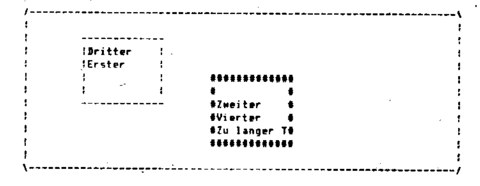


Bild 15 Der Rahmen begrenzt das Fenster, auch wenn der Text zu lang ist.

keln, die allen möglichen Ansprüchen gleichermaßen genügen. Hier soll nur eine Variante angeboten werden. Jeder Programmierer kann sich dann selbst schreiben, was für das jeweils von ihm zu lösende Problem passend ist. Die folgende Funktion holeRin bewirkt die Ausgabe eines Nutzertextes auf die im Fenster eingefügte Leerzeile und dahinter das Einlesen einer Real-Zahl. Das Abfangen von Fehleingaben wurde selbstverständlich ebenfalls gleich in der Funktion selbst realisiert.
Für die Nutzung der Funktion holeRin wollen wir uns wieder ein kleines Anwendungsbeispiel ansehen:

```
...
writeUp(1,'Maße fuer');
writeUp(1,'Kreiszyl.');
Ra:=holeRin(1,'Radius');
Ho:=holeRin(1,'Höhe');
...
```

Wenn wir annehmen, daß sich der Bediener einmal vertippt hat, so ergibt sich das in Bild 16 dargestellte Bildschirmaussehen. Dabei soll nicht verschwiegen werden, daß die angegebene Funktion keinen Schutz davor bietet, daß der Bediener aus dem Fenster hinausschreibt. Einen Schutz vor dem Überschreiben des Fensterbereiches bei der Eingabe von Daten durch den Nutzer gewährleistet zum Beispiel die Prozedur ReadWriteInWindow (Bild 14). Die erfordert aber eine etwas vom Gewohnten abweichende Eingabeführung und ist somit nicht überall anwendbar.
Nun fehlen uns noch einfache Hilfsmittel, die den Inhalt eines Fensters oder ein komplettes Fenster mit Rahmen vom Bildschirm löschen. Um den Aufwand hierfür so gering wie möglich zu belassen, nutzen wir wieder schon vorhandene Prozeduren. Die Prozedur windowclear (Bild 17) schiebt den betreffenden Fensterinhalt nach unten hinaus, indem sie die dafür erforderliche Anzahl von Leerzeilen in das Fenster einfügt. Die Nutzung der Prozedur InWindowDown für diese Aufgabe ist willkürlich gewählt. Mit der Prozedur InWindowUp könnte man den Fensterinhalt auch nach oben wegschieben. Die in Bild 17 angegebene Prozedur killwindow ist nicht zwingend notwendig. Ihre Funktion in der hier angegebenen Form basiert komplett auf bereits vorgestellten Komponenten der Fenstertechnik. Zuerst erfolgt die logische Vergrößerung des Fensters, welches vom Bildschirm gelöscht werden soll. Das neu definierte Fenster enthält nun auch den Rahmen des ursprünglich definierten. Somit entspricht das Löschen des neuen Fensterinhalts dem Löschen des alten Fensters mit Rahmen.

Verschieben von Fenstern

Wir haben uns nun alle Prozeduren geschaffen, um mit den fest deklarierten Fenstern zu arbeiten. Es fehlen nur noch Hilfsmittel, um bereits deklarierte und gezeichnete Fenster logisch und auf dem Bildschirm zu verschieben. Diese vier Prozeduren sind alle so geschrieben, daß die Fenster, an den Rand des Bildschirmes gefahren, immer kleiner werden. Konflikte treten erst dann auf, wenn die Fenster nur noch eine Zeile hoch oder eine Spalte breit sind. Um diese Konflikte zu vermeiden, sind äußere Maßnahmen erforderlich. Das Wandern der Fenster auf dem Bildschirm wird mit den Prozeduren in den Bildern 18 und 19 angewiesen. Dabei wurde, wenn möglich, wieder auf bereits vorhandenes zurückgegriffen. Das Wirkprinzip der Prozeduren windowtief und windowhoch (Bild 18) ähnelt dem der Prozedur kollwindow. Es wird dabei davon ausgegangen, daß das Verschieben eines kompletten Fensters nach unten oder nach oben genau dem Verschieben eines Fensterinhaltes mit korrigierten Koordinaten äquivalent ist. Nach dem Verschieben des Fensters auf dem Bildschirm wird dann die Berichtigung der neuen gültigen Eckpunkt-Werte im array wins vorgenommen.
Für das Wandern der Fenster nach rechts bzw. nach links war es aber notwendig, wieder **InLine-Maschinen-Code** zu nutzen, da ja keine Prozeduren für ein horizontales Verschieben von Fensterinhalten vorgesehen sind. Der innere Aufbau der Verschiebe-Prozeduren windowrechts und windowlinks (Bild 19) entspricht vom Prinzip her wieder

```
/------------------------------------------------------------\
!                                                            !
!        -------------                                       !
!        !Kreiszyl.   !                                      !
!        !Radius= 5,6!                                       !
!        !Radius= 5.6!       ##############                  !
!        !Hoehe = 3   !      #            #                  !
!        -------------       #Zweiter     #                  !
!                            #Vierter     #                  !
!                            #Zu langer T#                   !
!                            ##############                  !
!                                                            !
\------------------------------------------------------------/
```

Bild 16 Bildschirm nach der Eingabe in ein Fenster

```
procedure windowclear(i:integer);
  var z:integer;
  begin
    write(korsoroff);
    for z:=0 to wins[i].y2-wins[i].y1 do
      InWindowDown(i);
    write(kursoron)
  end;


procedure killwindow(i:integer);
  begin
    windowdef(i,wins[i].x1-1,wins[i].y1-1,wins[i].x2+1,wins[i].y2+1,' ');
    windowclear(i)
  end;
```

Bild 17 Lösch-Prozeduren

```
procedure windowtief(i:integer);
  begin
    if wins[i].y2=23 then wins[i].y2:=22;
    inslinedownin(wins[i].x1-1,wins[i].y1-1,wins[i].x2+1,wins[i].y2+2);
    windowdef(i,wins[i].x1,wins[i].y1+1,wins[i].x2,wins[i].y2+1,wins[i].c);
    if wins[i].y2=23 then windowwrite(i)
  end;


procedure windowhoch(i:integer);
  begin
    if wins[i].y1=2 then wins[i].y1:=3;
    inslineupin(wins[i].x1-1,wins[i].y1-2,wins[i].x2+1,wins[i].y2+1);
    windowdef(i,wins[i].x1,wins[i].y1-1,wins[i].x2,wins[i].y2-1,wins[i].c);
    if wins[i].y1=2 then windowwrite(i)
  end;
```

Bild 18 Vertikales Verschieben von Fenstern

Bild 19 Horizontales Verschieben von Fenstern

```
procedure windowrechts(i:integer);
  var v,h,z,j:integer;
  begin
    if wins[i].x2=79 then wins[i].x2:=78;
    z:=wins[i].x2-wins[i].x1+3;
    for j:=wins[i].y1-1 to wins[i].y2+1 do
      begin
        v:=(j-1)*80+wins[i].x2;h:=v+1;
        inline($21/$00/$f8/  (* LD HL,BS-Anfang *)
               $ed/$4b/h/    (* LD BC,(h)       *)
               $09/          (* ADD HL,BC       *)
               $eb/          (* EX DE,HL        *)
               $21/$00/$f8/  (* LD HL,BS-Anfang *)
               $ed/$4b/v/    (* LD BC,(v)       *)
               $09/          (* ADD HL,BC       *)
               $ed/$4b/z/    (* LD BC,(z)       *)
               $ed/$b8);     (* LDDR            *)
      end;
    for j:=wins[i].y1-1 to wins[i].y2+1 do
      begin
        gotoxy(wins[i].x1-1,j);write(' ')
      end;
    wins[i].x1:=wins[i].x1+1;wins[i].x2:=wins[i].x2+1;
    if wins[i].x2=79 then windowwrite(i)
  end;

procedure windowlinks(i:integer);
  var v,h,z,j:integer;
  begin
    if wins[i].x1=2 then wins[i].x1:=3;
    z:=wins[i].x2-wins[i].x1+3;
    for j:=wins[i].y1-1 to wins[i].y2+1 do
      begin
        v:=(j-1)*80+wins[i].x1-3;h:=v+1;
        inline($21/$00/$f8/  (* LD HL,BS-Anfang *)
               $ed/$4b/v/    (* LD BC,(v)       *)
               $09/          (* ADD HL,BC       *)
               $eb/          (* EX DE,HL        *)
               $21/$00/$f8/  (* LD HL,BS-Anfang *)
               $ed/$4b/h/    (* LD BC,(h)       *)
               $09/          (* ADD HL,BC       *)
               $ed/$4b/z/    (* LD BC,(z)       *)
               $ed/$b0);     (* LDIR            *)
      end;
    for j:=wins[i].y1-1 to wins[i].y2+1 do
      begin
        gotoxy(wins[i].x2+1,j);write(' ')
      end:
    wins[i].x1:=wins[i].x1-1;wins[i].x2:=wins[i].x2-1 ;
    if wins[i].x1=2 tHen windowwrite(i)
  end;
```

den am Anfang vorgestellten Prozeduren inslinedwonin und inslineupin. Schließlich ist das horizontale Verschieben ja auch wieder nichts weiter als ein Einfügen einer Leerspalte mit Verschiebung des restlichen Fensterinhalts. Auch diese beiden Prozeduren bewirken neben dem Verschieben des kompletten Fensters auf dem Bildschirm wieder eine Korrektur der Eintragungen im Array wins.

Beispiele für Fenstertechnik

Insgesamt wurden zwanzig Prozeduren bzw. Funktionen der Fenstertechnik entwickelt. Will man nur die Prozeduren inslinedown, inslineup oder inslinedownbis nutzen, so genügt es völlig, nur diese in ein Programm aufzunehmen. Anders sieht es damit aus, wenn man horizontal und vertikal begrenzte Fenster nutzen will. Dann sollten diese Lösungen auch das Array wins und die Prozedur windowdef enthalten. Da die meisten Prozeduren auf untergeordnete zurückgreifen, muß jeweils ermittelt werden, wo das Ende der Verschachtelung ist. Will man beispielsweise die Prozedur WriteDown nutzen, so muß das Programm auch die Prozeduren InWindowDown und inslinedownin beinhalten. Die für ein spezielles Programm benötigten Komponenten sollten dann in einem Include-File zusammengefaßt werden. In diesem kann man dann die teilweise sehr reichhaltigen Kommentare des Quelltextes löschen und die optische Strukturierung der Prozeduren auf ein Minimum reduzieren. Diese beiden Maßnahmen dienen der Verringerung von Quelltextgröße und Kompilierzeit.
Als Beispiel für die Anwendung der vorgestellten Fenstertechnik folgt jetzt ein Auszug aus einem Programm, in dem die Masse von geometrisch bestimmten Körpern berechnet wird (Bild 20). Da es notwendig ist, die jeweilige Dichte des entsprechenden Werkstoffes einzugeben, werden im Fenster mit der Nummer 1 unter anderem verschiedene Materialien angeboten, die mit jeweils einem Kennbuchstaben ausgewählt werden können. Die Anzahl der angebotenen Materialien kann aber die Zeilenzahl des Bildschirms überschreiten; es wurde eine Fenster-Lösung erforderlich. Die Funktion dieser Auswahlmenü-Lösung ist folgendermaßen:
Die ersten 19 Materialien werden in dem Fenster mit ihren jeweiligen Kennungen angeboten. Ist das vom Nutzer benötigte Material nicht dabei, erhält er durch Betätigung der Entertaste die nächsten 19 angeboten. Die Auswahl eines Materials erfolgt durch die Eingabe seines Kennbuchstabens.
Weiterhin verdient die Funktion lesen im Programmbeispiel Beachtung. Sie entspricht fast völlig der vorgeschlagenen Funktion holeRin unserer Fenstertechnik. Dadurch, daß in ihr das Fenster, in dem gelesen werden soll, fest eingestellt ist, wird zentral geregelt, in welchem Bereich des Bildschirmes die Eingabe von Zahlen erfolgen soll. Durch diese Festlegungen ist es gelungen, auf dem Bildschirm thematisch getrennte Bereiche einfach festzulegen und zu bedienen.
Zur Veranschaulichung des Programms ist in Bild 21 das Aussehen des Bildschirmes dargestellt. Die Pfeile geben dabei die Bewegungsrichtung der Fensterinhalte beim Rollen an. Es folgt noch ein weiteres Beispiel, dieses Mal vor allem für die Anwendung der Prozeduren zum Verschieben der Fenster auf dem Bildschirm (Bild 22). Das Demonstrationsbeispiel realisiert das dreimalige Umkreisen des Bildschirm-Mittelpunktes mit einem Fenster entgegen dem Uhrzeigersinn. Nach der Aufwärtsbewegung des Fensters wird der optische Effekt erzeugt, daß der Inhalt des Fensters auf Grund der Trägheit noch weiter nach oben steigt, und dann erst im Fenster herunterfällt.

```
program Massenberechnung;
type string80=string[80];
     suchtyp = array[1..40]of record
                 txt:string[40];kenn:char;wert:real end;
var wahl:char;
    Dichten:suchtyp;
    masse,volumen,dichte:real;
    wins :array[1..5] of record x1,y1,x2,y2:byte;c:char end;
{$I windows.inc }
procedure Bild;
  begin
    clrscr;
    windowdef(1,2,2,39,21,'!');windowwrite(1);   (* Menue  Fenster        *)
    windowdef(2,41,2,79,14,'!');windowwrite(2);  (* Kontrol/Eingabe Fenster*)
    windowdef(3,41,16,78,23,'!');windowwrite(3); (* Ergebnis  Fenster     *)
    gotoxy(1,23);write('Wahl Buchstabe ! ET fuer weiter');
    writeDown(3,'Faktor:              !Masse:           ');
    writeDown(3,'Aufrechnungs-        !Total-');
    writeDown(3,'---------------------!-------------------');
    writeDown(3,'Volum:               !Volum:');
    writeDown(3,'Letztes Einzel-      !Total-');
    writeDown(3,'---------------------!-------------------');
    writeDown(3,'Dicht:               !');
    writeDown(3,'letzte Einzel-       !');
  end;
function lesen(textt:string80):real;
  var zeil:string80;i:integer;r:real;
  begin
    repeat
      writeUp(2,textt+' = ');     (* Mit dieser function wird  *)
      read(zeil);val(zeil,r,i);   (* realisiert, dasz alle     *)
    until i =0;                   (* Zahleneingaben im Fenster *)
    lesen:=r;                     (* Nr. 2 erfolgen            *)
  end;
procedure kennhol(M:Suchtyp;var kenn:char;var dichte:real);
  var i:integer;
  begin
    windowclear(1);                            (* Da das Fenster Nr.1 *)
    i:=1;Dichte:=0.0;                          (* auch fuer andere    *)
    repeat                                     (* Menues genutzt wird *)
      repeat                                   (* wird das Menue als  *)
        if m[i].txt='' then i:=1;              (* Parameter uebergeben*)
        writeDown(1,m[i].txt+'   '+m[i].kenn);(* Es werden 19 neue   *)
        i:=i+1                                 (* Zeilen ins Menue    *)
      until i=19;                              (* Fenster geschoben   *)
      read(kbd,kenn);(* Nutzerabfrage *)
    until kenn<>#13; (* Neue Menuezeilen bei Betaetigung der ET-Taste *)
    i:=0;
    repeat  i:=i+1                              (* Suchen ausgewaehlter  *)
      until (m[i].kenn=kenn)or(m[i].txt='');    (* MenuePosition im Feld *)
    if m[i].txt<>'' then
      begin                    (* Wenn ausgewaehlte Position existiert:  *)
        writeUp(2,m[i].txt);   (* Auswahl ins Kontroll-Fenster schreiben *)
        dichte:=m[i].wert      (* Ausgewaehlten Wert uebernehmen         *)
  end end;
begin    (* Hauptprogramm *)
  Bild;
  ...........(* Fuellen des Feldes Dichten und Ermittelung des Volumens *)
  kennhol(Dichten,wahl,dichte);
  if (upcase(wahl)='F')or(dichte=0.0) then dichte:=lesen('Direkt-Wert');
  gotoxy(49,17);write(dichte:10:4);
  masse:=Volumen*dichte;
  gotoxy(67,23);write(masse:12:3)
end.
```

Bild 20 Programm Masse

```
-------------------------------------------------------------------------------
! Silber        S                   !                                         !
! Quecksilber   q          !        !                     /^\                 !
! Quarzglas     Q          !        !                      !                  !
! Porzellan     p          !        !                      !                  !
! Platin        P          !        !                      !                  !
! Nickel        n         \v/       !                      !                  !
! Natrium       N                   !                                         !
! Marmor        M                   !Radius = 1                               !
! Kobalt        k                   !V = 12                                   !
! Konstantan    K                   !Direkt-Wert = 12                         !
! Gold          G                   !Aluminium                                !
! Kork          c                   !Hardcopy                                 !
! Chrom         C                   ---------------------------------------
! Barium        b                   !letzte Einzel-     !                    !
! Blei          B                   !Dicht:     6.69000 !                    !
! Antimon       a                   !-------------------!--------------------!
! Aluminium     A                   !Letztes Einzel-    !Total-              !
! Direkt-Wert   F/f                 !Volum:   12.000000 !Volum:    48.566371!
!                                   !-------------------!--------------------!
------------------------------------Aufrechnungs-       !Total-              !
Wahl Buchstabe ! ET fuer weiter     !Faktor: 3          !Masse:    84.069019!
                                    ---------------------------------------
```

Bild 21 Bildschirmdarstellung des Masse-Programms

```
program FensterLauf;          (* Beispiel fuer Fensterverschieben *)
type string80 = string[80];
var i,j:integer;
{$I windows.ovl}
begin
  write(kursoroff);
  windowdef(1,10,5,20,10,'!');
  windowwrite(1);
  for j:=1 to 3 do
    begin
      for i:=1 to 10 do windowtief(1);   (* Fallen des Fensters           *)
      for i:=1 to 30 do windowrechts(1);(* Fenster wandert nach rechts   *)
      for i:=1 to 10 do windowhoch(1);   (* Steigen des Fensters          *)
      for i:=1 to 4 do inwindowup(1);    (* Weitersteigen  Fensterinhalt *)
      for i:=1 to 4 do inwindowdown(1);  (* Herunterfallen Fensterinhalt *)
      for i:=1 to 30 do windowlinks(1);  (* Fenster wandert nach links   *)
    end;
  killwindow(1);
  write(kursoron)
end.
```

Bild 22 Programm FensterLauf

wird fortgesetzt

Diskrete Fouriertransformation und digitale Signalverarbeitung

Dr. Jürgen Förster
Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik

Dr.-Ing. Jürgen Förster (31) studierte von 1978–1983 an der Technischen Hochschule Karl-Marx-Stadt, Sektion Automatisierungstechnik, und promovierte 1986 im Rahmen eines Forschungsstudiums zu Fragen der vibroakustischen Diagnostik. Seit 1986 ist er Assistent und seit 1987 Oberassistent im Wissenschaftsbereich Steuerungstechnik/Prozeßautomatisierung der Technischen Universität Karl-Marx-Stadt und arbeitet auf dem Gebiet der digitalen Signalanalyse.

Einführung

Die Diskrete Fouriertransformation (DFT) stellt eine der wichtigsten Basistransformationen der digitalen Signalverarbeitung dar. Sie und ihre Rücktransformation verknüpfen Signale im Zeitbereich mit Kennfunktionen wie Leistungsspektren, Autokorrelationsfunktionen, Kohärenzfunktion, Cepstrum usw.

Die Einführung der schnellen Fouriertransformation (FFT) als effektive Methode zur Berechnung der DFT durch Cooley und Tukey im Jahre 1965 ermöglichte den Einzug der Spektralanalyse in verschiedenste Gebiete von Wissenschaft und Technik. Als Beispiele seien hier die Akustik, die Schwingungsmeßtechnik, die Sprachanalyse, die Maschinendiagnose, die Bildverarbeitung und die Prozeßüberwachung genannt. Dennoch stellt die FFT einen bezüglich der Rechenzeit aufwendigen Algorithmus dar. Die jeweilige Aufgabenstellung bestimmt deshalb die Art und Weise der Realisierung ausgehend von Universalrechnern bis hin zu speziellen Hardwarekonfigurationen für Echtzeitaufgaben.

Ziel des Beitrages ist ein Vergleich dieser verschiedenen Möglichkeiten, um für gegebene ingenieurtechnische Aufgabenstellungen eine geeingnete Auswahl zu gestatten.

Als Realisierungsmöglichkeiten werden Universalprozessoren, Arrayprozessoren, Signalprozessoren und die Transputertechnik miteinander verglichen.

Algorithmen zur Berechnung der DFT

Mittels der DFT erfolgt eine Zerlegung eines finiten komplexen Signals in komplexe Fourierkoeffizienten:

$$X(k) = \frac{1}{N} \cdot \sum_{n=0}^{N-1} x(n) \cdot \exp(-j2\pi kn/N) \qquad n,k = 0\ldots N-1 \qquad (1)$$

Wird davon ausgegangen, daß eine komplexe Multiplikation mittels 4 reeller Multiplikationen (M_r) und 2 reeler Additionen (A_r) berechnet wird, so ergibt sich für die Gesamtzahl arithmetischer Operationen:

$$M_r = 4\,N^2 \qquad (2)$$

$$A_r = 4\,N^2 - 2N \qquad (3)$$

Dieser enorme Rechenaufwand führte zur Entwicklung immer effektiverer Algorithmen, angefangen bei den Basis-2-, Basis-4- und Basis-8-FFT-Algorithmen (mit 1, 2 oder 3 verschiedenen Butterfly-Operationen (BF)) über den Primfaktor-FFT-Algorithmus (PFA) bis zur Winograd-Fouriertransformation (WFTA). Einen Vergleich auf der Basis notwendiger arithmetischer Operationen sowie von Rechenzeitabschätzungen für Programme auf dem digitalen Signalprozessor TMS 32010 vermittelt /1/.

Eine aussichtsreiche Weiterentwicklung stellen die Split-Radix-Algorithmen dar /30/, /31/.

Weiterhin wurden auf der Basis aller genannten Verfahren spezielle Algorithmen zur Verarbeitung reeller bzw. konjugiert komplexer Eingangsdaten entwickelt, um die auftretende Redundanz in (1) zu beseitigen (/32/, /33/ sowie Schnelle Hartley-Transformation /2/, /3/).

Universalrechner und Universalprozessoren

Für viele Zwecke ist die Off-line-Spektralanalyse auf **Universalrechnern** oder auf **Universalprozessoren** ausreichend (Tafel 1).

Um auch Echtzeitaufgaben lösen zu können, wurden spezielle Hardwaremodule, auch **FFT-Prozessoren** genannt, entwickelt. Diese werden an einen Hostrechner gekoppelt und führen eine FFT mit wesentlich höherer Geschwindigkeit aus (z. B. FPR /4/ am System K 1520).

Für die moderne Signalverarbeitung ist es von Interesse, neben der FFT auch weitere Vektor- und Matrixoperationen mit hoher Geschwindigkeit ausführen zu können /5/. Für diese Verarbeitung von *Arrays* von Daten wurden **Arrayprozessoren** entwickelt. Es handelt sich um Multiprozessorsysteme, wobei Datenfluß und arithmetische Verarbeitung gleichzeitig erfolgen /6/. Sie erhöhen das Leistungsvermögen des als Hostrechner angeschlossenen Systems um ein Vielfaches (siehe Tafel 1). Von Vorteil gegenüber den reinen FFT-Prozessoren ist die Tatsache, daß auch die Weiterverarbeitung des Spektrums durch Divisions- und Quadratwurzeleinheiten sehr effektiv möglich ist.

Signalprozessoren

Die digitale Signalverarbeitung ist vorwiegend durch Algorithmen wie Skalarprodukt zweier Vektoren bei FIR- und IIR-Filtern, Butterfly-Operationen und Transformationen vom FFT-Typ gekennzeichnet. Diese lassen sich auf die Operationen Addition, Multiplikation mit Konstanten oder Variablen und Multiplikation mit anschließender Akkumulation zurückführen. Aus diesen Anforderungen heraus und mit den Möglichkeiten der VLSI-Technologie wurden folgerichtig die Signalprozessoren entwickelt, welche diese Operationen hardwareseitig unterstützen. Um 1980 marktwirksam geworden, ist derzeitig ein Ende der raschen Weiterentwicklung dieser Prozessoren nicht abzusehen. Eine umfassende Darstellung des internationalen Angebots vermittelt /10/.

Signalprozessoren sind durch Zahlendarstellungen mit 16-Bit- oder 24-Bit-Festkomma gekennzeichnet. Die neuesten Typen (μPD 77230, WE DSP32, DSP 96000, TMS 320C30) besitzen bereits Gleitkomma-Darstellung /27/.

Im Gegensatz zu Universalprozessoren wird die Havardstruktur oder die modifizierte Havardstruktur, das heißt die Trennung von Daten- und Adreßbus genutzt. Signalprozessoren arbeiten zur Erhöhung des Datendurchsatzes mit bis zu fünf Daten- und drei Adreßbussen (DSP 96000).

Die Arithmetikeinheiten bestehen fast immer aus einem schnellen Multiplizierer und einem Addierer bzw. einer Arithmetik-Logik-Einheit. Durch eine *Pipeline-Verarbeitung* kann die Verarbeitungsgeschwindigkeit zusätzlich erhöht werden. Für Skalierungsaufgaben ermöglichen *Barrel-Shifter* ein schnelles Bitverschieben um mehrere Stellen. Die Zykluszeiten konnten ständig verringert werden und

***Tafel 1** Rechenzeit für komplexe FFT auf Universalrechnern bzw. -prozessoren und Zusatzkomponenten (in ms)*

System	Prozessor	Takt	N					Bemerkungen
		MHz	64	128	256	512	1024	
16-Bit-Arithmetik								
K 1520	Z 80	2,5					<10000	
	UB8002	4,0	26	57	124	274		/7/
IBM PC	8088	4,77	19,0		110		610	/8/ mit 8087
Eagle-Turbo	8086	8,0	11,0		55		310	/8/
IBM PC/AT	80286	6,0	5,4		33		180	/8/
Gleitkommaarithmetik								
MicroVAX							45000	/9/
PDP 11			60		295		1400	/1/ 3 BF Basis-2
16-Bit-FFT-Prozessoren								
FPR (mit DMA auf K 1520-Bus)					5	10		/4/ mit reellen Eingangsdaten
Arrayprozessoren mit Gleitkommaarithmetik								
EC 1055-Matrix-Modul							13,8	/4/
ST-100							0,864	/6/
MicroVAX + Array-prozessor AP 500							4,7	/9/

betragen bei einigen Typen bereits weniger als 100 ns.

Leistungsfähige Signalprozessoren sind in der Lage, neben den arithmetischen Operationen parallel Adreßberechnungen durchzuführen. So besitzt beispielsweise der TMS 320C25 zur indirekten Adressierung 8 Auxilary-Register und ein zusätzliches Rechenwerk, das die Indexberechnung bei der FFT übernehmen kann. Die zusätzliche Möglichkeit des *Bitreversing* trägt ebenfalls zur schnelleren Berechnung der FFT bei.

Abschließend seien noch die Multiprozessorfähigkeit sowie die seriellen Schnittstellen zur Kommunikation mit Hostrechnern moderner Signalprozessoren erwähnt.

Einen Vergleich der Leistungsfähigkeit verschiedener Signalprozessoren ermöglicht Tafel 2. Für den TMS 32010 liegen für die verschiedenen FFT-Algorithmen umfangreiche Rechenzeitangaben vor /1/. Primfaktor- und Basis-8-FFT sind erwartungsgemäß die kürzesten Algorithmen. In /11/ wird zur Adreßrechnung für Daten und Koeffizienten ein Adreßsequenzer (AM 29540) verwendet. Er führt bei N = 1024 zu einer Einsparung von etwa 10 ms.

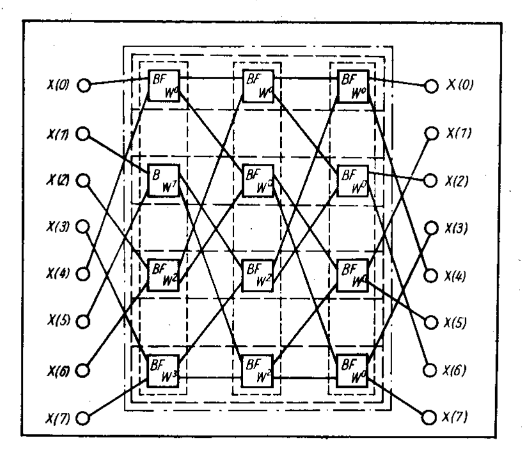
Interessant ist eine Anordnung mit zwei kaskadierten TMS 32010 /12/. Dabei übernimmt der erste Prozessor das *Windowing* der Eingangsdaten sowie 16 nacheinander abzuarbeitende FFTs der Länge N = 64. Entsprechend dem verwendeten Decimation-in-time-Algorithmus sind diese FFTs völlig identisch und können direkt programmiert werden. Die Rechenzeit für den ersten Prozessor beträgt 6,11 ms. Der zweite Prozessor führt 64 FFTs der Länge N = 16 mit verschiedenen Twiddlefaktoren in 8,5 ms aus und bestimmt damit die Gesamtgeschwindigkeit. Notwendig sind noch zwei durch Hardware ausgeführte Umordnungen der Daten nach dem ersten und nach dem zweiten Prozessor.

Neben der Leistungsfähigkeit eines solchen auf eine FFT zugeschnittenen Systems werden zugleich die Probleme der Multiprozessorsysteme deutlich. Unbedingt notwendig ist eine ausgewogene Bilanzierung der Rechenzeiten zwischen den Prozessoren. Zusätzliche Operationen wie Bitreversing oder Bildung des Leistungsspektrums können die Umverteilung von Programmteilen oder die Hinzunahme weiterer Prozessoren erfordern. Zusätzlich ist ein geeignetes Konzept der Buszuschaltung bzw. der Speicherumschaltung zu entwickeln. Die Leistungssteigerung wird also mit einer geringen Flexibilität gegenüber Einprozessorsystemen erkauft.

Gleitkommaprozessoren stellen die neueste Entwicklung der Signalprozessortechnik dar /20/, /21/, /27/. Dabei liegen die Transformationszeiten bereits unter dem Bereich der schnellen 16-Bit-Prozessoren.

Tafel 2 Rechenzeit für komplexe FFT auf ausgewählten Signalprozessoren (in ms)

System	N					Bemerkungen
	64/ 63	128	256	512/ 504	1024/ 1008	
16-Bit-Arithmetik						
TMS 32010 /1/	2,87				69,4	Basis-2, 1 BF
	1,92				45,3	Basis-4, 1 BF
	1,72				42,3	Basis-4, 3 BF
	0,6					Basis-4, 3 BF, direkt programmiert
	1,41			16,0		Basis-8, 2 BF
	1,53			17,4		PFA
	0,83					PFA, direkt programmiert
TMS 32010 /11/					59	Basis-2, 1 BF + Window-Operation
2 TMS 32010 kaskadiert /12/					8,5	Basis-2 + Window-Operation, reelle Eingangswerte, kein Bitreserving
TMS 32010 /13/			6,9			
TMS 320C25 /14/			1,5			Basis-4, direkt programmiert
			3,4			Basis-2
ADSP-2100 /15/					7,2	
DSP 56000 /16/, /17/					5,0	24 Bit Festkomma
LM 32900 /18/			1,49			Basis-4, direkt programmiert
			2,90		13,42	Basis-2
TS 68931 /19/	0,265		2,0			
ZR 34161 /28/					3,3	
GK-Arithmetik (8-Bit-Exponent, 24-Bit-Mantisse)						
µPD 77230 /20/				4,7	10,75	
DSP 32 /21/		2,0	4,3	9,0	19,2	Basis-2
DSP 96000 /27/					<2,0	
TMS 320C30 /27/					1,67	



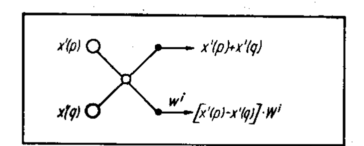


Bild 2 Butterfly-Operation Basis-2 mit Twiddlefaktor

Bild 1 Verschiedene Transputerstrukturen für FFT (N = 8, Basis-2)
- ——— ***Struktur 1: 1 Transputer pro BF***
- - - - - ***Struktur 2: 1 Transputer pro Spalte***
- — — ***Struktur 3: 1 Transputer pro Zeile***
- ····· ***Struktur 4: 1 Transputer pro FFT***

Transputerkonzept

Als Beispiel des Trends, anspruchsvolle Aufgaben mittels Multiprozessorsystemen zu lösen, wurde im vorangegangenen Abschnitt eine Kaskade mit zwei TMS 32010 vorgestellt /12/. Verschiedene Autoren weisen auf die Tatsache hin, daß die Leistung eines derartigen Systems nicht proportional zur Anzahl der Prozessoren zunimmt, sondern, bedingt durch Probleme der Aufgabenverteilung, der Kommunikation (Busbelastung) und des Aufwandes zur Ablauforganisation, ab einer bestimmten Menge wieder sinkt /5/, /23/, /24/.

Ein neues Niveau wird durch die gemeinsame Entwicklung von Hard- und Software mittels Transputern (und der Programmiersprache OCCAM) erreicht. Beispielsweise ist der Transputer IMS T 424 von Inmos durch eine 32-Bit-CPU mit einer Zykluszeit von 50 ns, einer Leistungsfähigkeit von 10 MIPS und 4KByte-on-chip-RAM mit einer Zugriffszeit von 35 ns gekennzeichnet. Weiterhin besitzt er einen integrierten Zeitgeber, eine leistungsfähige 8-Bit-Peripherieschnittstelle (4 MByte), eine 32-Bit-Speicherschnittstelle (25 MByte/s) und als wichtigstes vier serielle, bidirektionale Kommunikationskanäle (maximal 1,5 MByte/s pro Richtung) /24/. Damit lassen sich aus Transputern Matrixstrukturen mit Punkt-zu-Punkt-Verbindungen ohne konventionelle Bussysteme aufbauen. Zusätzlich ergeben sich Vereinfachungen bezüglich der notwendigen externen Logik und des Layouts von Leiterplatten.

OCCAM dient der Beschreibung paralleler Prozesse /22/. Grundelement der Sprache ist der Prozeß, der über Kommunikationskanäle mit anderen Prozessen Nachrichten austauschen kann. Weitere Möglichkeiten sind Eingabe, Ausgabe und Interrupt /23/.

Die Implementierung von OCCAM-Programmen ist sowohl auf einem einzelnen Transputer als auch auf einem Netzwerk mehrerer Transputer möglich. Damit kann beispielsweise ein FFT-Programm entsprechend den jeweiligen Forderungen nach Rechenzeit und Kosten auf verschiedene Strukturen umgesetzt werden. Bild 1 stellt einen FFT-Algorithmus mit Butterfly-Operationen (BF) dar. Die BF können als parallele Prozesse mit angeschlossenen Kommunikationskanälen betrachtet werden. Nutzt man den Transputer T 414 als *Butterfly-Maschine* (Bild 2), so wird eine BF in zirka 20 µs abgearbeitet /25/. In /25/ werden vier verschiedene Strukturen untersucht (siehe Bild 3), deren Leistungsfähigkeit Tafel 3 darstellt.

Tafel 3 ***Anzahl der Transputer (T), Zeit für eine FFT (t) und mögliche Abtastfrequenz (fs) für verschiedene Transputerstrukturen /25/***

N	Struktur 1 1 Transputer/BF			Struktur 2 1 Transputer/Spalte			Struktur 3 1 Transputer/Zeile			Struktur 4 1 Transputer/FFT		
	T	t/ ms	fs/ MHz	T	t/ ms	fs/ kHz	T	t/ ms	fs/ kHz	T	t/ ms	fs/ kHz
8	12	0,02	0,4	3	0,08	100	4	0,062	128	1	0,25	32
16	32	0,02	0,8	4	0,16	100	8	0,08	200	1	0,67	24
32	80	0,02	1,6	5	0,32	100	16	0,1	320	1	1,60	20
64	192	0,02	3,2	6	0,64	100	32	0,125	512	1	3,85	16
128	448	0,02	6,4	7	1,28	100	64	0,143	900	1	9,01	14
256	1024	0,02	12,8	8	2,56	100	128	0,166	1500	1	20,8	12,5
1024				10	13,8	80				1	103	9,7
4096				12	55,5	80				1	500	8,0

Die schnellste Transformationszeit ist erreichbar, wenn jede BF durch einen Transputer berechnet wird. Für N = 256 wird damit im Echtzeitbetrieb eine sonst kaum erreichbare Abtastfrequenz von 12,8 MHz möglich. Allerdings dürfte der Aufbau von 1024 Transputern (1 kW Verlustleistung /25/) einige Probleme mit sich bringen.
Die zweite Struktur, in der ein Transputer alle BF einer Spalte berechnet, hat etwa die Leistungsfähigkeit der modernsten Signalprozessoren (N = 1024, t = 13,8 ms) bei einem vertretbar erscheinenden Aufwand von 10 Transputern.
Struktur 3 ist durch eine zeilenweise Implementierung der BF gekennzeichnet und benötigt N/2 Transputer. Auch in ihrer Leistungsfähigkeit nimmt sie eine Stellung zwischen Struktur 1 und 2 ein (N = 256, fs = 1,5 MHz).
Wird der gesamte Algorithmus auf einen Transputer gelegt (Struktur 4), so ist diese Variante (N = 1024, t = 103 ms) immer noch schneller als beispielsweise der Universalprozessor I 80286 (Tafel 1).[1]
Aus diesen Ansätzen ergeben sich sofort neue Fragestellungen. Zunächst wäre zu untersuchen, ob sich zum Beispiel Split-Radix-Algorithmen ebenfalls für eine Transputerimplementierung eignen. Weiterhin ist von Interesse, ob sich Signalprozessoren in alternativer Weise als Transputer einsetzen lassen. Moderne Signalprozessoren verfügen über eine leistungsfähigere CPU und ausreichend On-chip-RAM. Die Fragestellung bezieht sich hier hauptsächlich auf die Fähigkeit, untereinander kommunizieren zu können. Schließlich wäre zu prüfen, ob sich für derartige Konfigurationen die Sprache OCCAM zur Programmierung eignet.

VLSI-Arrays von Prozessorelementen

Eine neue Qualitätsstufe wird durch die Integration ganzer ein- oder zweidimensionaler Felder von Prozessorelementen auf einem Chip erreicht. Aus den Besonderheiten der VLSI-Technologie leiten sich folgende Bedingungen für die Entwicklung ab /26/:
- Einschränkung auf lokale Kommunikationsverbindungen zwischen den Prozessorelementen
- asynchrone Arbeitsweise der Prozessorelemente untereinander
- modulare Struktur
- Programmierbarkeit der Prozessorelemente.

Als erste Vertreter wurden die *Systolischen Arrays* als Netzwerke identischer Prozessorelemente entwickelt, die beispielsweise zur Berechnung der Matrix-Multiplikation eingesetzt werden können. Da jedoch für den Datentransport und für die zeitaufwendigeren Rechenoperationen das gleiche Zeitlimit vorgesehen ist, ergeben sich Geschwindigkeitsverluste. Ein weiteres Problem ist der synchrone Datenfluß des gesamten Arrays. Diese Nachteile werden mit der asynchronen Arbeitsweise des *Wavefront-Arrays* überwunden /26/.
Während sich beispielsweise IIR-Filter in Strukturen mit ausschließlich lokalen Kommunikationsverbindungen umwandeln lassen, ist dies für FFT-Algorithmen in der herkömmlichen Form nicht möglich. Implementierungsmöglichkeiten ergeben sich jedoch für den ursprünglichen DFT-Algorithmus, der eine Matrix-Vektor-Multiplikation darstellt (1).

[1] Der T 800 als Transputer der 2. Generation berechnet eine reellwertige 32-Bit-FFT in 55 ms.

Literatur

/1/ Burrus, C. S.; Parks, T. W.: DFT/FFT and Convolution Algorithms. John Wiley & Sons, New York 1984
/2/ Bracewell, R. N.: The Fast Hartley Transform. Proceedings of the IEEE, Vol. 72, NO 8, Aug. 1984, S. 1010
/3/ Sorensen, H. V.; Jones, D. L.; Burrus, C. S.; Heidemann, M. T: On Computing the Discrete Hartley Transform. Transactions on Acoustics, Speech and Signal Processing, VOL. ASSP-33, NO 4, October 1985, S. 1231
/4/ Geissler, R.: FFT-Modul für K 1520, Radio, Ferns., Elektron., Berlin 34 (1985) 9, S. 588
/5/ Whitehouse, H. J.: Signal Processing Technology. Modern Signal Processing (Proceedings fo the Arab School on Science and Technology), Hemisphere Publishing Corporation 1985, S. 370
/6/ Runge, H.: Supercomputer-Leistung für Prozeßrechner. Elektronik (1986) 12, S. 69
/7/ Wogek, L.; Schewe, K.-P.: Schadensdiagnose mit spezieller FFT-Implementierung auf einem UB 8002-Arithmetikmodul. Heft SIGNALANALYSE der Wissenschaftlichen Schriftenreihe der TU Karl-Marx-Stadt (1986) 12, S. 31
/8/ Schindler, M: FFTs get personal on small computers. Electronic Design, May 2, 1985, S. 250
/9/ Dittberner, W.; Vieten, M.: Meßwerterfassung und -verarbeitung mit freiprogrammierbaren Rechnersystemen. Elektronik (1987) 2, S. 82
/10/ Gluth, R.: Integrierte Signalprozessoren. Elektronik (1986) 18, S. 112
/11/ Küng, R.: Flexibler FFT-Prozessor löst Echtzeitaufgaben. Elektronik (1986) 21, S. 101
/12/ Benetazzo, L.; Narduzzi, C.; Offelli, C.: Improving Spectral Analysis Systems with digital Signal Processors. presented at the LASTED International Symposium „Applied Signal Processing And Digital Filtering", Paris, June 19–21, 1985
/13/ Wiggins; Mcmahan; Tarrant; Gass: DSP boards help tackle a tough class of AI Tasks. Electronics, August 21, 1986, S. 64
/14/ Kropp, G.: CMOS-Signalprozessor für echte Parallelverarbeitung in Mehrprozessorsystemen. Elektronik (1986) 18, S. 159
/15/ Roesgen, J.; Tung, S.: Moving memory off chip, DSP μP squeezes in more computational power. Electronic Design, February 20. 1986, S. 131
/16/ Ein später Sieger? Elektronik (1986) 11, S. 34
/17/ Auderer, G.: Digitaler Signalprozessor mit RISC-Architektur. Elektronik (1986) 22, S. 210
/18/ Baskerville, G.; Schwartz, M.; Schiappacasse, J.: Digitaler Signalprozessor in CMOS-Technik. Elektronik (1986) 6, S. 57
/19/ Omenzetter, M.: Universeller Signalprozessor bietet hohen Datendurchsatz. Elektronik (1986) 4, S. 71
/20/ Eichen, B.; Davis, M. H.; Kulp, B. D.: Floating-point math integrated on chip makes DSP IC a standout. Electronic Design, February 20. 1986, S. 159
/21/ Ferro, F.: Einchip-Signalprozessor mit Fließkommaverarbeitung (Teil 2: Programmierung). Elektronik (1986) 19 S. 123
/22/ May, D.: OCCAM – Programmiersprache für den Systementwurf. Elektronik (1982) 22, S. 83
/23/ Eckelmann, P.: Transputer: Mikrorechner-Konzept für hohe Verarbeitungsleistung. Elektronik (1983) 24, S. 51
/24/ Eckelmann, P.: Architektur und Anwendung des Transputers. Elektronik (1984) 4, S. 59
/25/ Eckelmann, P.: Transputer – richtig eingesetzt, Beispiele für die Fourier-Transformation in OCCAM. Elektronik (1985) 4, S. 57
/26/ Kung, S. Y: VLSI Array Processor for Signal Processing. Modern Signal Processing (Proceedings of the Arab School on Science and Technology), Hemisphere Publishing Corporation 1985, S. 393
/27/ Milt, L.: FOCUS Digital Signal Processors. Electronic Design, October 13, 1988, S. 161
/28/ VSP Family Overview. Zorun Corporation, Santa Clara
/29/ Neumerkel, D.: Einsatz von Transputern in Spracherkennung und Robotik, Mikroprozessortechnik, Berlin 3 (1989) 1, S. 13
/30/ Duhamel, P.; Hollmann, H.: ‚Split Radix' FFT Algorithm. Electronics Letters 1984, Vol. 20, No. 1, S. 14
/31/ Sorensen, H. V.; Heidemann, M. T.; Burrus, C. S.: On Computing the Split-Radix FFT. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-34, No. 1, February 1986, S. 152
/32/ Duhamel, P.: Implementation of „Split-Radix" FFT Algorithms for Complex, Real, and Real-Symmetric Data. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-34, No. 2, April 1986, S. 285
/33/ Sorensen, H. V.; Jones, D. L.; Heidemann, M. T.; Burrus, C. S.: Real-Valued Fast Fourier Transform Algorithms. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol ASSP-35, No. 6, June 1987, S. 849

KONTAKT

Technische Universität Karl-Marx-Stadt, Sektion Automatisierungstechnik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 5613332

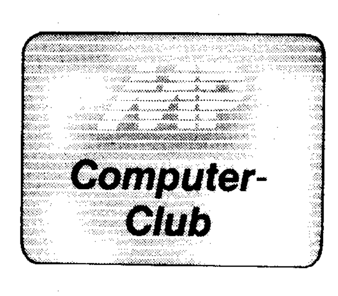
TERMINE

6. Computerfachtagung

WER? Bezirksverband der Kammer der Technik Frankfurt (Oder)
WANN? 25. und 26. Oktober 1989
WO? Frankfurt (Oder)
WAS?
- neuer Computer aus Mühlhausen
- Einchipmikrorechner
- Probleme bei der Softwareentwicklung
- Softwarepakete für 16-Bit-Rechner
- Ausstellung „Computer der volkseigenen Industrie, Eigenbaucomputer und Softwareentwicklung"

WIE? Meldungen für die Ausstellung richten Sie bitte bis zum 31. 8. 89 an Dr. Scheuschner, Ingenieurbetrieb Mikroelektronik Frankfurt (Oder), Tel. 369242. Nähere Informationen sind beim Bezirksverband der KDT Frankfurt (Oder), Ebertusstraße 2, Frankfurt, 1200; Tel. 369360, erhältlich.

Georgi/Dr. Scheuschner



Schnelle Algorithmen für Geraden und Kreise in Forth

Die im folgenden beschriebenen Algorithmen wurden aus den für die jeweilige Kurve geltenden Differenzgleichungen entwickelt. Bei einer Geraden gilt:

$$\Delta Y_n = k \cdot \Delta X_n$$

Ein angenäherter Kreis entsteht durch Lösung des Systems

$$\Delta X_n = kY_n; \Delta Y_n = -kX_n$$

Diese Gleichungen wurden nun so umgeformt, daß jede Rechnung die Koordinaten eines neuen Punktes liefert. Dann ergibt sich für die Gerade:

$$X_{n+1} = X_n + 1$$
$$Y_{n+1} = Y_n + k$$

und für den Kreis

$$X_{n+1} = X_n + 1$$
$$Y_{n+1} = Y_n - X_n/Y_n.$$

Mit diesen Formeln erhält man einen geschlossenen Linienzug, wenn $|\Delta Y/\Delta X| \leqq 1$ ist, anderenfalls sind X und Y zu vertauschen.
Da die Anfangswerte und auch die Koordinaten der zu berechnenden Punkte ganze Zahlen sind, wurde die Division durch eine modulo-Arithmetik ersetzt, wie sie im Prinzip schon bei den digitalen Differentialanalysatoren angewendet wurde. Der maximale Fehler der berechneten Koordinaten beträgt ein Pixel, wie man durch Vergleich mit den exakten Werten zeigen kann.
Die Programmbeispiele in Basic sollen lediglich das Verständnis der Zusammenhänge erleichtern.
Durch das Programm in Bild 1 wird eine Gerade vom Ursprung zu einem einzugebenden Endpunkt gezeichnet. YL ist hier und in den folgenden Beispielen der zu Y gehörende niederwertige Teil, in dem Änderungen akkumuliert werden. Eine entsprechende Routine in Maschinenkode (U 880) kann durch Vereinfachung des Programms für den Kreis erhalten werden (Bild 3).
Die Programme in den Bildern 2 und 5 zeichnen einen Achtelkreis um den Ursprung des Koordinatensystems.

```
10  INPUT X1,Y1
20  LET X=0 : LET Y=0 : LET YL=Y1
30  PLOT X,Y
40  LET X=X+1 : LET YL=YL+Y1
50  IF X>200 THEN STOP
60  IF YL<X1 THEN GOTO 30
70  LET YL=YL-X1 : LET Y=Y+1 : GOTO 30
```

Bild 1 Gerade in Basic

```
10  INPUT R
20  LET X=0 : LET Y=R : LET YL=0
30  PLOT X,Y
40  LET X=X+1 : LET YL=YL+X
50  IF X>Y THEN STOP
60  IF YL<Y THEN GOTO 30
70  LET YL=YL-Y : LET Y=Y-1 : GOTO 30
```

Bild 2 1/8-Kreis in Basic

```
INIT:    LD E,0            ;X:=0
         LD A,(RAD)
         LD D,A            ;Y:=R
         XOR A             ;YL:=0
ANFANG:  PUSH AF
         CALL PLOT         ;Punkt
         POP AF
         INC E             ;X:=X+1
         ADD A,E           ;YL:=YL+X
         JR C,UEB          ;Ueberlauf
         CP D
         JR C,TEST
UEB:       SUB D           ;YL:=YL-Y
           DEC D           ;Y:=Y-1
TEST:    LD L,A
         LD A,D
         CP E              ;X>Y ?
         LD A,L
         JR NC,ANFANG
         RET
```

Bild 3 1/8-Kreis in U 880-Maschinenkode

```
100 VARIABLE RADIUS
0 VARIABLE X 0 VARIABLE Y 0 VARIABLE YL
: INIT 0 X ! RADIUS @ Y ! 0 YL ! ;
: DELTA 1 X +! X @ YL @ + DUP Y @ >=
        IF Y @ - -1 Y +! ENDIF
  YL ! ;
: ENDE? X @ Y @ > ;
: PKT X @ Y @ PLOT ;
: KREIS 8 INIT BEGIN PKT DELTA ENDE? UNTIL ;
```

Bild 4 1/8-Kreis in Forth

```
: PKT  2DUP PLOT ;
: ENDE?  2DUP >  ;
: DELTA  SWAP 1+ ROT OVER +
         ROT  2DUP >=
         IF SWAP OVER - SWAP 1 - ENDIF
         ROT SWAP ;

: 1/8-KREIS  ( R --- )
         0 0 ROT
         BEGIN PKT DELTA ENDE? UNTIL
         DROP DROP DROP ;
```

Bild 5 Forth-Programm ohne Variable

Um einen Vollkreis in beliebiger Lage zu erhalten, sind die Symmetriebedingungen am Kreis auszunutzen und die Koordinaten des Mittelpunktes zu addieren.
Im Forth-Programm in Bild 4 wurden der besseren Anschaulichkeit wegen Variablen verwendet. Durch INIT erhalten sie ihre Anfangswerte. Im Wort DELTA werden die neuen Werte für X und Y errechnet (dementsprechend die Zeilen 40, 60 und 70 des Basic-Programms). Mit ENDE? wird die Abbruchbedingung geprüft, es bleibt ein Flag auf dem Stack. In der Begin-Until-Schleife des letzten Wortes werden durch PKT so lange Punkte mit den jeweils neu berechneten Koordinaten gesetzt, bis dieses Flag wahr ist.
Bei der Variante in Bild 5 befinden sich die Variablen ständig auf dem Parameterstack.
Zur Initialisierung ist vor Aufruf von 1/8-KREIS der Radius einzugeben, der gleich als Anfangswert für Y verwendet wird. X und YL werden auf 0 gesetzt. Durch ROT kommt Y auf dem Stack an die erste Stelle, dann folgen X und YL. In dieser Reihenfolge liegen die Variablen vor und nach Ausführung von PLOT und DELTA auf dem Stack. Bei ENDE? kommt an oberster Stelle noch das über den Abbruch entscheidene Flag hinzu. Nach UNTIL werden die nicht mehr benötigten Werte vom Stack entfernt. Das Programm ist schwerer lesbar, aber um den Faktor 1,25 schneller als das Beispiel in Bild 4. Dieses Verhältnis ist jedoch in starkem Maße von der Laufzeit der Routine PLOT und damit von der Implementierung und der zugrunde liegenden Hardware abhängig.
In dem zum Vergleich benutzten Rechner mit U 880 und 4 MHz Taktfrequenz dauerte die Berechnung der Koordinaten eines Kreispunktes nach dem Algorithmus in Maschinenkode im Mittel 20 Mikrosekunden, eine Ausgabe auf den Bildschirm beanspruchte jedoch 300 Mikrosekunden. Der Basic-Befehl CIRCLE (auch eine Maschinenkoderoutine) benötigte für die Ausgabe eines Kreispunktes etwa 4 ms. Demgegenüber sind sogar die Forth-Programme mit etwa 2 ms noch schneller. Das Basic-Programm in Bild 1 war dagegen, nicht zuletzt wegen der Gleitkomma-Rechenoperation, mit 22 ms deutlich langsamer.
Abschließend soll darauf hingewiesen werden, daß nach dieser Methode auch andere Kurven, wie Parabeln oder Exponentialfunktionen dargestellt werden können.

Peter Taege

BASICODE im Rundfunk

Ab September beginnt bei „REM – das Computermagazin" des Schulfunks von Radio DDR II ein neuer Kurs, diesmal zum Programmieren in BASICODE.
BASICODE ist ein in den Niederlanden entwickelter universeller Übertragungscode, der einheitlich für verschiedene Rechner einlesbar und abarbeitbar ist und damit den Austausch von Programmen verschiedener Computertypen ermöglicht.
Entsprechende Übersetzungsprogramme existieren bereits für die KC-Typen, für AC 1, Z 1013, C 64, Atari und Sinclair Spectrum.
Start des neuen Kurses ist der 6. September 1989, 17.00 Uhr; die weiteren Folgen dann – wie bekannt – jeden 2. Mittwoch um 17.00 Uhr im Rahmen des Schulfunkprogramms.
Zu dem Kurs wird ein kostenloses Begleitmaterial bereitgestellt, das unter folgender Anschrift angefordert werden kann, sofern ein frankierter und mit der eigenen Adresse versehener A5-Umschlag mitgeschickt wird:
REM – Schulfunk Radio DDR II,
Kennwort BASICODE,
Nalepastraße, Berlin, 1160

Dr. Baumann

KC 85/3 als Eingabeterminal für PC 1715, BC A 5120

Dr. Helge-Karsten Klink,
Christian Kompart
VEB Chemiewerk Kapen

Mit der Erweiterung des Modulsortiments für den KC 85/2 oder /3 durch die Module M012 (Texor) und M003 (V.24) gewann der Kleincomputer eine zunehmende Bedeutung für die Textverarbeitung.
Da jedoch nicht für jeden in der Textverarbeitung eingesetzten KC zur Textausgabe ein Druckgerät zur Verfügung steht und somit die bereits genannten Module in ihren Möglichkeiten nicht voll genutzt werden können, wurde eine hardwaretechnische Lösung dieses Problems gefunden und eine Kopplung zwischen dem KC und PC oder BC hergestellt.
Die auf dem KC im Textverarbeitungssystem Texor erstellten Texte können nun direkt in den PC bzw. BC überspielt werden. Dabei werden keine neuen Treiberroutinen auf dem KC benötigt, sondern gleich die im Texor-Modul befindlichen Druckerroutinen verwendet. So werden alle ASCII-Zeichen mit einer Geschwindigkeit von 9 600 Baud übertragen.
Nach der Übertragung werden die in den PC oder BC überspielten Texte von den mitübertragenen Druckersteuerzeichen mittels Transformationsprogramm gesäubert.
Die Nutzung dieser Kopplung beschränkt sich natürlich keineswegs auf die Textverarbeitung.
So können beispielsweise Programme (in dBase, Algol, Basic, usw.) oder dBase-Dateien im Texor des KC erstellt werden, die dann anschließend auf dem PC/BC lauffähig sind.
Die maximale Dateigröße im Texor des KC beträgt etwa 15 KByte. Die Übertragungsentfernung beträgt bei speziell geschirmtem Kabel (Koax) vom KC zum PC/BC maximal 80 m.

Hardwaretechnische Voraussetzungen

- KC 85/2 oder 3
- Modul M012 (Texor)
- Modul M003 (V.24)
- PC 1715 oder BC A 5120 mit V.24-Interface
- Kopplungsleitung (vom Anwender herzustellen)

Softwaretechnische Voraussetzungen

- Telekommunikationsprogramm TLCM, hergestellt vom VEB Büromaschinenwerk Sömmerda, in der Version 1.1 (Version 0.6 realisiert die Kopplung nur bei PC 1715)
- Transformationsprogramme zur Text-, Programm- und Dateibearbei-

tung der auf PC 1715 oder BC A 5120 überspielten Texte

Realsierung der Kopplung

Am KC 85/2 oder /3 wird der Kanal 1 des Modul M003 (V.24) genutzt. Der Modul M003 sollte dabei im Modulschacht 8 stecken. Beim PC 1715 und BC A 5120 wird das eingebaute V.24-Interface genutzt.
Zur Übertragung der Daten vom KC zum PC oder BC genügt ein Stereo-überspielkabel, an dem an der PC/BC-Steckseite eine 13polige Steckleiste angelötet wird.

- Masse = Betriebserde (A9 = B8 (Brücke))
- T × D = Sendedaten
- CTS = Sendebereitschaft
- DTR = Betriebsbereitschaft
- R × D = Empfangsdaten

Transformationsprogramme

Um die mitüberspielten Druckersteuerzeichen, die für die weitere Verarbeitung der vom KC überspielten Texte, Programme oder Dateien in den PC/BC hinderlich sind, zu eliminieren, wurden dazu 2 Beispielprogramme in dBase geschrieben.

Programm TRANS

Das Programm TRANS eliminiert die Druckersteuerzeichen innerhalb einer überspielten Programm- oder Textdatei und bereitet diese so auf, daß diese danach ein normales dBase- bzw. Textformat hat.
Der überarbeitete Text kann anschließend im jeweiligen Textprozessor des PC/BC abgearbeitet werden. Handelt es sich um ein Programm,

Signal-V.24 KC	Stecker KC	Stecker PC/BC	Signal-V.24 PC/BC
F102 Masse	2	A1	F102 Masse
F103 R × D	(br) 1	A3	F103 T × D
F104 T × D	(ge) 3	B4	F104 R × D
F106 DTR	(we) 5	B6	F106 CTS
F108 CTS	(gn) 4	B8, A9	F108 DTR
			F109 DCD

Initialisierung des KC und PC/BC

Das Programm TLCM.COM in den PC (SCP/5.1) oder BC (SCP/1.5) laden.
Anschließend wird die Parameterdatei TLCM.PAR geladen.
Parameter:

Device:	PC 1715 oder (BC A 5120/30 – K6028/8025-A)
Mode:	Full-Duplex
Speed:	9 600 bd
Parity:	No
Stop-Bits:	1
Bit/Zeichen:	8
Prozedur:	DTR
CR-Echo:	CR

- Beginn des Datenempfangsmodus mit ta (TALK)
- Eröffnung der Empfangsdatei im PC/BC durch ESC-G
- Eingabe des Namens der Empfangsdatei
- Eingabe der Übertragungsgeschwindigkeit n
- Abspeicherung nach Ende der Übertragung durch ESC-E
- Abbruch des Programms TLCM durch e

Der KC wird über das im Texor-Modul befindliche INIT-Menü initialisiert.
Parameter:

Steuerzeichen	→ N
Geräteauswahl	→ K
Schriftart	→ K
Sperrschrift	→ 1
Zeilenabstand	→ 1
Anfangsspace	→ Enter

- Beginn der Übertragung im Textmodus (Texor) durch e (Flattersatz)
- Hinter Zeilen/Seite: die maximale Zeilenanzahl des Textes, des Programms oder der Datei eingeben → ET.
- Die Abfrage Druck J/N: mit J beantworten
- Abbruch der Übertragung jederzeit mit BRK
- Ende der Übertragung wird durch 'Befehl' angezeigt!

beispielsweise in dBase, kann dieses nach der Abspeicherung durch TRANS abgearbeitet werden. Handelt es sich um eine Textdatei, die im Textprozessor des PC/BC weiterbearbeitet oder ausgedruckt werden soll, gibt man in die letzte Zeile ein: quit to 'TP'
Nach der Abarbeitung der überspielten Dateien im TRANS wird nun automatisch der Textprozessor im PC/BC gestartet.

Programm TRANS1

Das Programm TRANS1 eliminiert die Druckersteuerzeichen innerhalb einer vom KC zum PC/BC überspielten und zuvor auf KC erstellten dBase-Datei und bereitet diese im PC/BC so auf, daß sie dann als dBase-Datei in einem dBase-Programm im PC/BC verarbeitet werden kann.
Ein Beispiel für TRANS1 geschrieben in dBase siehe im Bild 2.
Die fertiggestellte dBase Datei kann nun im PC/BC in entsprechenden Programmen weiterbearbeitet werden.
Es ist erkennbar, daß die Druckersteuerzeichen, wie Leerzeichen und anderes, durch entsprechende dBase Anweisungen eliminiert werden.
Jeder Nutzer des KC als dezentrales Datenerfassungs- und Eingabegerät für einen PC 1715 oder BC A 5120 kann und muß natürlich selber das Format seiner auf dem KC zu erstellenden Datei, ob Datenbank, Text oder Programm, festlegen und kann die hier nur als Beispiel aufgeführten kurzen dBase Transformationsprogramme für seine Zwecke verändern oder erweitern. Auf jeden Fall ist mit der Möglichkeit dieser Kopplungsmethode ein durchaus sinnvoller Einsatz des KC garantiert.
Nebenbei erhöht sich die Anzahl der Anwender von Computertechnik in vielen Betriebsbereichen und der Auslastungsgrad der im Einsatz befindlichen PC/BC unter dem Gesichtspunkt der Nutzung dieser als zentrale Druckstationen.

```
erase
set talk off
store                    to d,d1
@ 10,10 say 'Dateinamen vollstaendig eingeben (mit Punkt)'
@ 2,2 say 'Welche Datei bearbeiten ?' get d
@ 4,2 say 'Name der Zieldatei ?      get d1
read
use struc
copy stru to dat
use dat
append from &d1 sdf
use
delete file dat
quit
```

Bild 1 Programm TRANS

Bild 2 Programm TRANS1 ▼

```
erase
set talk off
@ 10,10 say 'Dateinamen vollstaendig eingeben (mit Punkt)'
@ store '                  to d,d1
@ 2,2 say 'Name der aufzunehmenden Datei' get d
@ 4,2 say 'Name der Ergebnisdatei         get d1
read
use struc1
copy structure to dat
if.not.file ('&d1')
copy structure to &d1
endif
use dat
append from &d delimited
use
use dat
delete
delete all for z1='          '.or.z3=0
pack
use &d1
append from dat
delete file dat
use
?'Datei','&d1','fertiggestellt !!'
```

Die Nutzungszeiten von PC/BC für die Programmierung oder Erstellung großer Dateien sind ebenfalls verkürzbar. Ein Test dazu ist jedoch unmöglich, da der KC sich im Textprozessorsystem Texor befindet, was ein kleiner, jedoch nicht unbedingt hemmender Nachteil ist.
In unserem Betrieb wird diese Kopplung in der Rechenstation angewendet. Dadurch kann der Programmierer etwa 75 Prozent seiner Programme, die er im Algol schreibt, auf KC erstellen. Zuvor führte das zu einer Blockierung des BC A 5120, was wiederum zur Blockierung des mit dem BC gekoppelten KRS 4201 führte. Jetzt hingegen ist der BC für etwa 75 Prozent der Eingaben zum KRS freigesetzt, der Programmierer benötigt lediglich noch die Zeit zum Test und zum Abschluß seines Programms auf dem BC, bevor er dieses in den KRS überspielt.
In diesem Sinne viel Erfolg bei der Anwendung dieser Kopplung.

☒ KONTAKT ☏

VEB Chemiewerk Kapen, PSF 158, Dessau, 4500; Tel. 72 44 App. 415

Kleines Lexikon der Mikrorechentechnik



L
wie Lochstreifenstanzer und Lochstreifenleser

Zeichnung: Dahmen



Börse

Tourenoptimierung mit BC-TOUR, Version 2.0

BC-TOUR ist ein Softwarepaket zur Lösung von Tourenproblemen mit Hilfe von 8-Bit und 16-Bit-Mikrorechentechnik. Es kann sowohl als Anschlußprojekt an die ESER-Softwarepakete DISKO-2 und DISKO-TOUR als auch separat genutzt werden. Bei der Anwendung von BC-TOUR können folgende Bedingungen berücksichtigt werden:
– Bedarf und Aufnahmebereitschaft der Kunden (Erfassung des Bedarfs in max. 10 Liefermengeneinheiten möglich)
– Be- und Entladezeiten
– feste Aufenthaltszeiten bei den Kunden
– Anzahl und Kapazität der Fahrzeugtypen
– Einsatzdauer der Fahrzeugtypen (Berücksichtigung von 2 Schichten möglich)
– Beschränkung der Tourdauer
– Optimierung nach zwei Maßeinheiten möglich (z. B. Masse und Volumen)
– Geschwindigkeit (Berücksichtigung von territorialen Besonderheiten ist möglich, z. B. Bahnübergänge, Fähren usw.)
– als Entfernungsmatrix wird nur eine Matrix der direkten Entfernungen benötigt.
BC-TOUR berücksichtigt folgende maximalen Problemgrößen, die zum Teil auf Wunsch des Nutzers vom Entwickler erweitert werden können: 500 Kunden, 200 Zonen, 50 Touren, 50 Fahrzeuge, 20 Fahrzeugtypen, 2000 direkte Entfernungen. BC-TOUR ist voll menügesteuert. Die Funktionen der Kursorsteuerung und die Syntax der Nutzerantworten wurden von Textprozessor, Turbo-Pascal und Dienstprogrammen übernommen, so daß der PC-geschulte Anwender keine Einarbeitung benötigt.
Geliefert werden Programme und Anwenderdokumentation auf nutzereigenen Disketten. Garantie- und Änderungsdienst sowie Anlaufunterstützung werden gewährt.

VEB Datenverarbeitungszentrum Potsdam, Dortustraße 46, Potsdam, 1500; Tel. 39311 oder 39312 *Käske/Miers*

Forth-Spracherweiterung für Künstliche Intelligenz

Auf der Basis des Backtrackprinzips wird eine vollständige Symbiose algorithmischer und deklarativer Programmiermethoden in einem gemeinsamen Sprachkonzept realisiert. Weiterhin wird die strategisch gesteuerte Lösungssuche für Aufgaben großer Suchtiefe unterstützt. Es existieren folgende Komponenten:
Mechanismus für unbestimmte Verzweigungen (TRY) und Backtracking (FAIL). Dies bedeutet, daß ein Programm Verzweigungen enthalten kann, deren Entscheidungskriterien erst im nachhinein gewonnen werden.
Universell und sehr einfach anwendbare Strategiefunktionen anhand anwenderspezifischer Bewertungskriterien oder fuzzylogischer Verknüpfung heuristischer Wichtungen.
Das **Note-and-Follow-Konzept**. Es besteht darin, den gültigen (d. h. zum Erfolg führenden) Programmweg zu notieren (NOTE), sozusagen über die durchlaufenen Programmverzweigungen und ausgeführten Backtrackings buchzuführen, um dann erneut (ggf. mehrfach) diesem notierten Weg durch das Programm backtrack- und verzweigungsfrei zu folgen (FOLLOW). Dies dient einer erheblichen Programmvereinfachung. Sehr zweckmäßig erscheint auch der unmittelbare Einsatz zur Maschinen- und Robotersteuerung, indem das gleiche Programm zur Simulation und zur direkten Steuerung benutzt wird.
Expertensystem-Tools: Wissensorganisation und Symbole, Fakten und Regeln, Expert-Dialog-Ebene (insbesondere für Spezialfachsprachen).
Die Spracherweiterung ist Bestandteil des NILES-FORTH-Entwicklungssystems (siehe MP 11/1988, S. 346); der Preis beträgt 9500,– M. Bitte fordern Sie ein ausführliches Informationsblatt oder eine DEMO-Diskette (Schutzgebühr: 185,– M) an.

VEB Werkzeugmaschinenkombinat „7. Oktober", Abt. TP6, Gehringstraße 39, Berlin, 1120; Tel. 3631641 *Noack*

Dateikonvertierung UNIX – MS-DOS

Im Rahmen einer Installation des hierarchischen Datenbankbetriebssystems HIDA durch die Firma startex GmbH, Bonn (BRD), beim VEB E. A.-Seemann-Verlag Leipzig, Redaktion Allgemeines Künstlerlexikon, wurden zwei Utilities für die Dateikonvertierung geschaffen. Das Programm **tar2dos** konvertiert Dateien im tar-Format (Betriebssystem WEGA oder UNIX-kompatibles Betriebssystem) in Dateien im MS-DOS-Format. Das Programm **w2crlf** ändert in Textdateien die Zeilenendmarkierung von 0AH (WEGA bzw. UNIX) in 0D0AH (MS-DOS). Beide Utilities laufen unter MS-DOS oder kompatiblen Betriebssystemen.
Eine kostenlose Weitergabe an Interessenten ist unter der Kontaktadresse möglich (bitte MS-DOS-Diskette, 720 KByte, einsenden).

VEB E. A. Seemann-Verlag, Redaktion Allgemeines Künstlerlexikon, Jacobstraße 6, PSF 846, Leipzig, 7010 *Schmidt*

Turbo-Pascal-Routinen für dBase III-Zugriff

In unserem Institut wurde ein unter dem Namen **TPDBASE** zusammengefaßter Satz von Routinen entwickelt, mit deren Hilfe ein Turbo-Pascal-Anwenderprogramm dBase III- bzw. dBase III Plus-Datenbankdateien verarbeiten kann. Die Routinen sind unter MS-DOS und kompatiblen Betriebssystemen und der Programmiersprache Turbo-Pascal, Version 4.0, anwendbar. TPDBASE hat folgenden Leistungsumfang:
– Lesen und Schreiben von dBase III-Datensätzen einschießlich Dateierweiterung
– Zugriff auf einzelne Feldwerte und Suchen nach speziellen Feldwerten
– Funktionen, die den dBase-Kommandos COPY STRUCTURE und PACK entsprechen
– Verfügbarkeit der Struktur der dBase III-Datei im Anwenderprogramm
– keine Beschränkung der Anzahl der gleichzeitig eröffneten dBase III-Dateien durch TPDBASE
– Möglichkeit der Auswahl der vom Anwenderprogramm tatsächlich benötigten Felder der dBase III-Datei. Anzahl und Anordnung der übrigen Felder beeinflussen das Anwenderprogramm nicht.
– wählbare interne Darstellung der Feldwerte
– Möglichkeit, einen Dateiausschnitt im Hauptspeicher zu führen. Dabei lädt TPDBASE nur die ausgewählten Felder in der gewünschten internen Darstellung. Der Zugriff zu den Daten wird dadurch sehr effektiv. Die Verwaltung dieses Hauptspeicherbereiches (Nachladen, Sichern) kann TPDBASE übertragen werden.

Bauakademie der DDR, Institut für Technologie und Mechanisierung, Plauener Straße 163/165, Berlin, 1092; Tel. 37833333 *Schimpf*

Interaktives Dialogsystem für P8000

IDS 8000 ist ein leistungsfähiges, dialogorientiertes Softwareprodukt zur Verwaltung von Nutzerprogrammen im UNIX-kompatiblen Betriebssystem WEGA. IDS 8000 ist für solche Anwender entwickelt worden, die mit ihrem Computer die vielfältigsten Probleme des betrieblichen Reproduktionsprozesses vorzugsweise durch eigene Dialogprogramme oder auch durch Nutzung anderer Softwarelösungen bearbeiten wollen. Alle unter IDS laufenden Programme können autonom ihre Aufgabe erfüllen. Es besteht die Möglichkeit, daß gleichzeitig unterschiedliche Problemgebiete wie Material, Absatz, Produktion usw. ohne gegenseitige Behinderung (gleichzeitige Manipulation von Dateien) aktiv sind. Um eine einfache Handhabung zu gewährleisten, wurde das System so konzipiert, daß auch Anwender, die keinerlei UNIX-Kenntnisse besitzen, Programme auswählen und abarbeiten können. Alle Systemteile von IDS 8000 wurden in der Programmiersprache C realisiert, wobei großer Wert auf Änderungsfreundlichkeit gelegt wurde. Zum Start von IDS 8000 wurden das UNIX-Interface, Login und Password genutzt. Nach Eingabe des Paßwortes erfolgt der Start aller notwendigen Systemprozesse (IDSBASE, TTYCTRL, CFM, evtl. TRANSFER) und der Beginn einer Dialogführung mit dem Nutzer des entsprechenden Terminals. Mit der Eingabe eines (ET) wird ein Menüprozeß aktiviert, und mit der Auswahl einer Menüfunktion kann das entsprechende Programm gestartet werden. Voraussetzung für die Entwicklung der Nutzerprogramme ist zur Zeit die Anwendung der Programmiersprache C. Für die Dateienarbeit verfügt IDS über eine umfangreiche Unterprogrammbibliothek, die sowohl sequentielle als auch indexsequentielle Dateimanipulation erlaubt. All diese Funktionen arbeiten in Wechselbeziehung mit der Systemkomponente CFM (Common-File-Management) oder auch mit möglichen Transferprozessen zur Vernetzung mehrerer P 8000.

VEB Schuhfabrik „Paul Schäfer" Erfurt, Klement-Gottwald-Straße 52, Erfurt, 5082; Tel. 382424 *Küffner/Herold*

Aufarbeiten von Daten auf Lochband

Für das Aufarbeiten von Datenbeständen, die auf Lochband gespeichert sind, wurde ein PC (unter CP/M) mit einer Lochbandleseeinrichtung gekoppelt. Der Einlesevorgang wird durch ein Turbo-Pascal-Programm mit Optionen gesteuert. Im Rechner bilden die eingelesenen Daten eine Textdatei, die variabel weiterverarbeitet werden kann. Als Hardwarebestandteile werden ein Lochbandleser ohne Steuereinheit sowie eine minimale (selbst aufzubauende) Steuerelektronik benötigt. Zum Aufbau der Rechnerkopplung sind ein freies PIO-Tor zur Dateneingabe sowie vier Steuerpins eines weiteren PIO-Tores des Rechners erforderlich. Folgende Funktionen sind möglich:
– Rechtslauf, Linkslauf, Stop des Lochbandlesers
– Erkennen von Bandende, Bandabriss
– Einlesen der Lochbandinformationen.
Der Bauelementeaufwand ist gering. Es werden vier LS-TTL-Schaltkreise, drei Transistoren (SF 128) und einige Transistoren zur Pegelanpassung benötigt.

Institut für Züchtungsforschung der AdL, Ethel-und-Julius-Rosenberg-Sraße 22/23, Quedlinburg, 4300; Tel. 470 *Boczek/Liedecke*

Software für Gaststätten

Zur Anwendung liegen Softwarelösungen für den C 128 D und für den PC 1715 unter einem CP/M-kompatiblen Betriebssystem vor. Sie beinhalten folgende Lösungen:
SPEIKA – Kalkulationsprogramm für Speisen und Getränke
URKON – Programm zur Planung, Abrechnung, Analyse und Kontrolle der Urlaubsinanspruchnahme
KRASTA – Programm zur Abrechnung und Analyse des Krankenstandes
KABRAN – Programm zur Kraftstoffabrechnung mit Monats-, Quartals- und Jahresabrechnung.
Alle Softwarelösungen sind anwenderfreundlich als Menüprogramme ausgelegt. Die verschiedenen Eingabedaten werden in Dateien abgelegt, die bei der Programmabarbeitung aufgerufen werden. Das Ausdrucken von aktuellen Auswertelisten ist möglich. Zum Ausdruck für SEIKOSHA-Drucker wurde außerdem ein Druckprogramm entwickelt, das vor dem Ausdruck geladen werden muß.

VE EHB (HO) Gaststätten Greifswald, Baderstraße 3, Greifswald, 2200; Tel. 76208 *Werth*

Konvertierungssystem für Leiterplattendaten

Der rechnergestützte Entwurf von Leiterplatten, das Erzeugen der Fertigungsunterlagen und die Fertigung selbst erfordern ein flexibles Konzept für die Umwandlung der nicht durchgängig standardisierten, herstellerspezifischen Datenformate der einzelnen Bearbeitungsanlagen. Für diesen Zweck wurde ein erweiterbares Datenkonvertierungssystem entwickelt. Es ist in Turbo-Pascal programmiert und damit portabel, setzt eine spezielle Overlaytechnik ein und ist modular aufgebaut. Zur Zeit existieren vier Modultypen: Dialog-, Verarbeitungs-, Eingabe- und Ausgabemodul. Die Eingabe- und die Ausgabemodule legen das Eingabe- und Ausgabeformat des jeweiligen Konvertierungslaufes fest. Diese Module sind als Overlay ausgelegt und beliebig kombinierbar. Die fest definierten Schnittstellen sowie die vom Bearbeitungsmodul gebotenen Leistungen gestatten es, ohne großen Aufwand weitere Eingabe- und Ausgabemodule zu entwickeln. Gegenwärtig sind als Eingabeformate u. a. ALZG-3, LM-B20, MHF, LKA, GERBER und als Ausgabeformate ALZG-3, LM-B20, MHF, LKA, EXCELLON bearbeitbar. Implementiert wurde das System auf einem 8-Bit-Rechner unter CP/M.

Technische Universität Dresden, ZWGB, Mommsenstraße 13, Dresden, 8027; Tel. 4579593

Dr. Drentschev

Anschluß VT 27000 an 8-Bit-Mikrorechner

Es wurde eine Lösung für den Anschluß des Videoton-Zeilendruckers VT 27000 an 8-Bit-Mikrorechner mit dem Betriebssystem SCP erarbeitet (Bürocomputer A 5120/30, PC 1715, K 1520). Damit stehen dem Nutzer die Leistungen des von der SM-4 her bekannten Schnelldruckers auch an 8-Bit-Arbeitsplätzen zur Verfügung. Geeignet ist die Lösung vor allem für den Druck längerer Dokumentationen. Hardwareseitig ist der Aufbau einer kleinen Zusatzschaltung erforderlich, der z. B. auf einer Universalleiterplatte erfolgen kann.
Zur Nachnutzung werden folgende Unterlagen bereitgestellt:
– Stromlaufplan der Zusatzschaltung
– Assemblerquelle für den BIOS-Treiber, der in das Betriebssystem einzubinden ist.

VEB Elektroprojekt und Anlagenbau Berlin, ZFT, Abt. RC4, Rhinstraße 100, Berlin, 1140; Tel. 5509531/App. 29 oder 31

Dr. Mertins/Löbel

Festplatte für EC 1834 und Software für Plotter

Die Nachrüstung des EC 1834 mit einer Harddisk für IBM-PCs konnte mit einer Änderung des ROM-BIOS auf den Festplattencontrollern erreicht werden. Für die EPROMs 2764 wurde als Ersatz eine kleine Emulatorplatine mit 4 × 2716 gebaut. Folgende Controller und die dazugehörigen Festplatten sind bisher auf diese Weise am EC 1834 betreibbar:
– WD 1002A-WX1 (Tandon 20 MByte)
– WD 1002A-27X (Wincard 30 MByte)
– OMTI 5527A (Seagate ST238R 30 MByte)
Für andere Systeme ist eine ähnliche Anpassung sicher möglich. Die durch den DMA-Schaltkreis 8057 reduzierte Blockgröße (maximal 16 KByte) bei der Datenübertragung hatte keine Auswirkungen unter DCP oder MS-DOS. Die zum EC 1834 mitgelieferten Serviceprogramme (FDISK, HDINIT, HDPARK) sind ebenfalls verwendbar.
Für den Kleinplotter XY4131 wurde ein Softwarepaket entwickelt, das eine weitgehende Kompatibilität zum HP 7475 garantiert und auf Bürocomputern lauffähig ist. Durch die Anpassung an den HP 7475 kann es für einen Großteil der Grafiksoftware, die auf 16-Bit-Rechnern zur Verfügung steht, verwendet werden. Es besteht aus folgenden Bestandteilen:
GETPLT – Empfang von Plotterkommandos über V.24 und Ablegen in einer Datei
HPXY4131 – Plotten von Dateien bzw. Plotterkommandos, die über die V.24 empfangen werden
PLINST – Installieren der Schnittstellen und der Standardplottereinstellungen.
Für den Plotteranschluß ist ein unbeschaltetes PIO-Port im Bürocomputer erforderlich. Im praktischen Betrieb sind folgende Kopplungen möglich:
PC —— V.24 ⟶ BC → Datei
PC — Diskette → BC → XY4131
PC —— V.24 ⟶ BC → XY4131

Forschungszentrum für Bodenfruchtbarkeit Müncheberg, Abt. Wissenschaftlicher Gerätebau, Wilhelm-Pieck-Straße 72, Müncheberg, 1278; Tel. 82241

Dr. Petzold

Anfertigen von Leiterplatten

Die Feinmechanische Werkstatt Reese bietet die Anfertigung von Leiterplatten (NDKL, ein- und zweiseitig) nach der vom Kunden gelieferten maßstabgerechten Vorlage (Leiterbildzeichnung) an.

Firma Dipl.-Ing. Thomas Reese, Feinmechanische Werkstatt, wissenschaftliche Geräte und Leiterplatten, Ambrosiusplatz 5, Magdeburg, 3014; Tel. 48752

Reese

Grafik-Druckerprogramm für K 6311 am K 8915

Für den Rechner K 8915 vom VEB Robotron-Rechenelektronik Meiningen/Zella-Mehlis wurde ein Druckertreiber entwickelt, der nachladbar (oder besser vorladbar) sich in den freien RAM-Bereich zwischen F000H und FFBH lädt und der den Ablauf normaler Programme sowie den zur Verfügung stehenden Speicherplatz für Nutzerprogramme nicht verringert. Das I/O-Byte wird auf User-List-Kanal umgeschaltet. Hardwaremäßig ist die ATSK 7028.20 (1 × IFSS, 1 × V.24) auszutauschen. Im Datenaustausch werden im Drucker DTR-Protokoll und 9600 Baud eingestellt. Auf Wunsch kann auch eine ASV K 9021 Verwendung finden, wobei aber auf der Platine die Erzeugung der -12-V-Stromversorgung entsprechend geändert werden muß. Die Adressen können auf Wunsch in den Programmen entsprechend gelinkt werden. Von der V.24-Schnittstelle werden die Leitungen Masse, DTR sowie Rx/TX benötigt. Im V.24-Steckverbinder der ATS sind die Leitungen 107 und 108 zu brücken. Die Funktionen Schriftgröße, Zeichensatz und Grafik-/Alpha-Modus werden menügesteuert aufgerufen. Mit einem Hardcopy-Programm können BWS-Inhalte der Grafikplatte VIS2A (AdW), 512 × 256 Pixel, ausgedruckt werden. Das Programm kann ebenfalls zur Verfügung gestellt werden.
Angebotene Leistungen sind eine Hardware-Dokumentation und eine Software-Beschreibung. Die Programme können wahlweise auf 8"- oder auf 5¼"-Disketten geliefert werden.

Technische Hochschule Ilmenau, Sektion Phyteb, PSF 327, Ilmenau, 6300; Tel. 74621 und 74681

Gleichmann/Behrens

Kopplung PC und SD 1156

Für die weitere Verwendung vorhandener Seriendrucker SD 1156 mit Schnittstelle SIF 1000 (TTL-Pegel) und die Nutzung der damit verbundenen Vorteile wurde eine Hard- und Softwarelösung geschaffen, die es erlaubt, die Drucker an fast beliebige BC/PC/AC mit V.24-Schnittstelle anzuschließen. Die Lösung besteht aus einer Leiterplatte mit EMR U 882, 2 KByte EPROM und 1 KByte RAM und dem entsprechenden Druckersteuerprogramm. Der EMR wird bei minimalem Verdrahtungsaufwand im Drukker installiert, ohne daß an der übrigen Druckertechnik etwas verändert werden muß (wichtig für die Wartung). Realisiert wird über den EMR der Datenaustausch mit dem Rechner über V.24-Protokoll und die vollständige Druckeransteuerung über das Interface SIF 1000. Am PC ist keine hard- oder softwareseitige Veränderung notwendig. Erprobt und eingesetzt wird die Lösung bei uns vorrangig für den PC 1715 am Rechnerausgang PRINTER.

VEB teltomat Teltow, Werk für Straßenbaumaschinen, Abt. LOE, Ruhlsdorfer Straße, Teltow, 1530

Schnabel

Druckprogramm RUND

Das Programm **RUND** verwaltet acht Adressendateien und unterstützt den Druck von Rundschreiben mit persönlichen Anreden sowie den Druck von Briefumschlägen mit dem Textverarbeitungssystem TP unter dem Menüpunkt Kombodruck. Der Druck von Anschriftenverzeichnissen ist möglich. Dem Nutzer stehen unter anderem die Funktionen Eingabe, Korrektur, Druck, Zeigen, Verschieben und Kopieren innerhalb der Adressendateien und die Auswahl von Anschriften für ein Rundschreiben mittels Kombodruck zur Verfügung. Neben der Anschrift, der Anrede und dem Namen ist das Speichern von Zusatzinformationen zu jeder Anschrift möglich (Telefonnummer usw.). Nutzer dieses Programms sind Betriebsleitungen, Genossenschaften und gesellschaftliche Organisationen, die gleichlautende Texte an eine große Anzahl von immer wiederkehrenden Adressaten versenden. Das Programm läuft unter den Betriebssystemen SCP und DCP. Die Dateien können unter dBase II bzw. dBase III weiterverarbeitet werden.
Bei Zusendung einer formatierten 5¼-Zoll-Diskette kann die Nutzerdokumentation angefordert werden.

Firma Ulrich Schmidt, Bahnhofstraße 17, Altlandsberg, 1274; Tel. 408

Schmidt

Modulbibliothek für Turbo-Pascal

Die entwickelte Turbo-Pascal-Modulbibliothek bietet insbesondere für Datenerfassungsprogramme folgende Vorteile:
– Gewährleistung einer einheitlichen Bedienoberfläche auf unterschiedlicher Technik (BC, PC 1715, K 8915, A 7100, A 7150, EC 1834 u. ä.)
– Bedienfreundlichkeit und Bedienkomfort
– rationelle Programmierung
– uneingeschränkte Portabilität von Programmen
– effektive Pflege- und Wartungsarbeiten.
Die Modulbibliothek besteht aus einzelnen Boxen, die über Includeanweisung in Turbo-Pascal-Programme eingebunden werden und folgendes realisieren:
EINGABE-BOX ermöglicht Bildschirmeingaben (real, integer, byte oder string), Vorwertübernahme, Standardwertübernahme, Löschen des Eingabefeldes u. ä. Durch Eingabe von ? lassen sich jederzeit Helptexte abfordern; Minimum- und Maximumprüfung sind möglich.
HELP-BOX ermöglicht die Nutzung von variablen Helptexten
KAPAZ-BOX zur Prüfung der freien Diskettenkapazität
KOPF-BOX zur einheitlichen Gestaltung des 1. Bildschirmbildes einschließlich Datumeingabe und -prüfung
MENÜ-BOX zur einheitlichen Darstellung der Funktionsauswahlbilder
MACHDBD-BOX zur Erzeugung des Dateidefinitionsblockes von Datenbankdateien
FILE-BOX für Prüfen, Einrichten, Fortschreiben u. ä. von Dateien
LISTE-BOX zur wahlweisen Ausgabe von Listen auf Drucker oder Diskette
BROS-BOX zur Verarbeitung von BROS-Disketten unter SCP
LIESKBD-BOX wandelt den Tastencode der Kursortasten am 16-Bit-Rechner in CTRL-Zeichen um
FTASTEN-BOX gibt zurück, welche Funktionstaste am 16-Bit-Rechner gedrückt wurde.
Zu jeder Box existiert eine umfangreiche Beschreibung.

Institut für Forstwissenschaften Eberswalde, Organisations- und Rechenzentrum Potsdam, Virchowstraße 39/41, Potsdam, 1591; Tel. 76941

Neumann

Wir suchen ...

... einen Treiber für den Anschluß einer Kassettenmagnetbandeinheit K 5261.01 an den A 7100.

Technische Universität Dresden, Sektion Forstwirtschaft, Bereich Forsteinrichtung und Forstliche Ertragskunde, Plenner Straße 8, PSF 10, Tharandt, 8223; Tel. 6231

Vogel

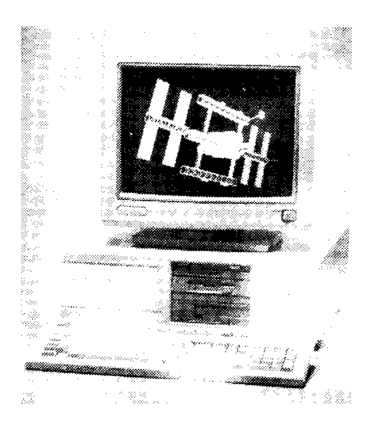


Entwicklungen und Tendenzen

Erste PC-Familie mit 80486-Prozessor

Von der englischen Firma Apricot Computers wurden Anfang Juni die ersten Personalcomputer vorgestellt, die Intels neuesten 32-Bit-Mikroprozessor, den 80486, verwenden.
Sie sind für den Einsatz der Betriebssysteme MS-DOS, OS/2 und UNIX vorgesehen. Das Einstiegsmodell VX 400/30 hat 4 MByte RAM, einen 128-KByte-Cache und eine 338-MByte-Festplatte. Bei 25 MHz Taktfrequenz wird eine Leistung von 15 MIPS erreicht. *MP*

Compaq Deskpro 386/33

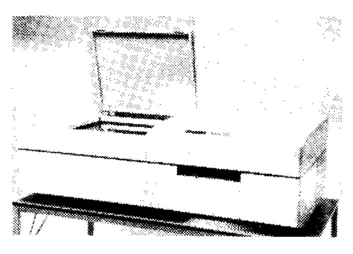


Zwar gelang es der Firma Compaq nicht, den ersten 386er-PC mit 33 MHz-Prozessor vorzustellen – siehe 4. Umschlagseite –, doch will man den Anspruch, Marktführer bei den 386er PCs zu sein, mit „dem ersten PC der Welt auf der Basis eines mit 33 MHz getakteten 80386-Prozessors, der sofort lieferbar ist" verteidigen. Ende Mai wurde das neue Modell präsentiert, das vor allem für anspruchsvolle Anwendungen wie CAD/CAE, Softwareentwicklung und Finanzanalysen sowie als Fileserver in Multiusersystemen ausgelegt ist. Durch die Kombination der bekannten Compaq-Flex-Architektur mit der höheren Taktrate soll eine um 35 Prozent höhere Leistung als bei 25-MHz-PCs möglich sein. Dazu tragen auch der 64-KByte-Cachespeicher bei sowie die Möglichkeit des (gleichzeitigen) Einsatzes der neuen Koprozessoren Intel 80387 und Weitek 3167 mit je 33 MHz. Die Flex-Architektur, die einen 32-Bit-Speicherbus beinhaltet, ermöglicht durch das Concurrent-Bus-Prinzip, bei dem Speicher- und Peripheriebus gleichzeitig angesprochen werden, eine gleichmäßige Auslastung aller Subsysteme. Standardmäßig ist der 386/33 mit 2 MByte RAM, einem 5,25-Zoll-Diskettenlaufwerk und einer Festplatte zwischen 84 und 650 MByte ausgestattet. Durch Nutzung des 32-Bit-Erweiterungssteckplatzes kann der RAM auf bis zu 16 MByte ausgebaut werden.
Daneben gibt es noch 7 freie Steckplätze im Industriestandard. Der Einbau von bis zu 5 Massenspeichern erlaubt es, die interne Speicherkapazität bis auf 1,3 Gigabyte zu erweitern – ein Wert, der nach Aussagen von Compaq bisher von keinem anderen Desktop-Computer erreicht wird. *MP*

EMS-Chip

Die Voraussetzung für effektives Multitasking sind große Speicher, von denen jeder Task (Aufgabe) ein Speicherbereich zugewiesen wird. Die Standards für Speichererweiterungen unter MS-DOS sind der EMS 4.0 bzw. der EEMS (Enhanced Expanded Memory Standard; siehe dazu MP 3/1989, Seite 89). Die USA-Firma Chips & Technologies kündigte Anfang des Jahres die Produktion des 82C631 für die Verwaltung des EMS-Speicher an. Der *EMS Mapper* eröffnet für die PC-Hersteller neue Möglichkeiten. Bisher wurden die Schaltkreise für die Speicherverwaltung auf den RAM-Erweiterungskarten – die mit 1-MBit-RAMs bereits bis zu 16 MByte umfassen – plaziert. Der 82C631 kann künftig auf der Hauptplatine untergebracht werden. Dadurch wird es möglich, die Kapazität der Erweitungskarten weiter zu steigern. Die 128 Register des 82C631 zum Ansteuern der EMS-Speicherbänke sollen bei Programmen wie Windows und Desqview zu erheblichen Geschwindigkeitssteigerungen führen. Zum Erzeugen oder Aktualisieren der Speicherfenster ist lediglich ein Ein-/Ausgabekommando nötig. *MP*

Hochauflösender Farbscanner ACS 100 Color



Als einen Meilenstein im farbigen Druckvorstufenbereich betrachtet die Firma Agfa-Compugraphic ihren neuen hochauflösenden Farbscanner. Der ACS 100 Color ist ein CCD-Flachbettscanner, der eine variable Auflösung von 300 bis 2400 Punkten pro Zoll (dpi) für Anwendungen in der farbigen Druckvorstufe bereitstellt. Der Einsatz der CCD-Technik (charge coupled device) und die hohe Präzision des Flachbett-Mechanismus sollen für ein sehr schnelles Arbeiten des Systems sorgen.
Agfa-Compugraphic wird den neuen Farbscanner in vorhandene und zukünftige Systemumgebungen integrieren; aufgrund seiner offenen Schnittstellen-Architektur steht der Scanner jedoch auch für andere Systemumgebungen zur Verfügung.
Einige Schlüsselfunktionen und besondere Leistungsmerkmale:
– Bei einer Pixeltiefe von 10 Bit pro Farbe sind 1024 Graustufen pro Farbe erreichbar.
– Scan-Geschwindigkeiten (Beispiele):
A3-Seite Halbtonabbildung bei 300 dpi Auflösung: 60 Sekunden; 35 mm-Dia bei 2400 dpi: 90 Sekunden
– SCSI-Bus zum Anschluß an offene Pre-Press-Systeme.
– Ethernat-Option zur schnellen Datenübertragung. Standard-Protokolle wie TCP/IP werden unterstützt. *MP*

Spezialprozessor für Künstliche Intelligenz

Einen Spezialprozessor für Aufgabenstellungen der Künstlichen Intelligenz, insbesondere für Symbolverarbeitung und Echtzeitanwendungen, hat die französische Firma Sodima entwickelt. Es handelt sich dabei um einen 32-Bit-Prozessor in RISC-Architektur (Reduced Instruction Set Computer). Der Prozessor wird in einem Gastrechner als schneller Coprozessor verwendet. Die Rechenleistung beträgt 10 bis 20 MIPS bei einer Taktfrequenz von 10 bis 20 MHz. Damit können in Verbindung mit dem Betriebssystem KOS (Knowledge-based Operating System) Aufgaben der Künstlichen Intelligenz effektiver gelöst werden. In Verbindung mit einem Rechner Sun 3-160 kann die Leistung durch den Coprozessor um den Faktor 18 erhöht werden. Der in Parameterform programmierbare Interferenzgenerator erreicht dadurch eine Abarbeitung von 3600 Regeln pro Sekunde.
Gegenwärtig ist der Prozessor als VLSI-Schaltung in CMOS-Technik mit einer Strukturbreite von 1,2 μm ausgelegt. Er umfaßt 50 000 Transistorfunktionen in einem Gehäuse mit 176 Anschlüssen. Die Firma Sodima sieht folgende Weiterentwicklung vor:
– Schaffung eines Parallelrechners mit 4 bis 8 Prozessoren
– Herstellung einer Version in Gallium-Arsenid-Technik mit einer Rechenleistung von 100 bis 200 MIPS und einer Taktfrequenz von 200 MHz.
Erste Anwendungen sollen für Echtzeitaufgaben in Flugkörpern und in Kernkraftwerken gefunden worden sein.

Quelle: Computer Zeitung. – Leinfelden-Echterdingen 20 (1989) 5. – S. 28 *Wi*

Integration bei elektronischen Bauelementen

Die Produktion von 1-MBit-DRAMS soll nach Ansicht von Vertretern der Firmen Siemens und Philips die Voraussetzung schaffen, um die Produktion der Bauelementekategorie im Submikrometerbereich zu beherrschen. Bei den 1-MBit-DRAMS zeichnet sich schon jetzt eine zweite Generation ab, die eine Zugriffszeit von 65 ns aufweist, was eine Linienbreite von 0,5 μm voraussetzt.
Es wird angenommen, daß in diesem Jahr die Linienbreite von 0,5 μm in der industriellen Produktion nicht mehr unterschritten wird. Das Hauptaugenmerk gilt der Senkung der Ausschußquote bei der bisher erreichten Linienbreite.
Obwohl für dieses Jahr eine größere Anzahl von IC-Typen mit Submikrometer-Geometrie erwartet wird, soll die Anzahl der ausgelieferten 4-MBit-DRAMs noch gering bleiben, da die Ausschußquote nur eine kostenungünstige Produktion zuläßt.
In diesem Jahr soll mit CMOS-ICs zu rechnen sein, die Taktraten von 50 bis 75 MHz aufweisen. Die Anzahl der Gates pro Logik-IC soll sich auf 200 000 bis 500 000 steigern lassen. Bipolare Logik-ICs sollen Gate-Zahlen von 20 000 bis 50 000 aufweisen, bei Taktraten von 2 bis 6 GHz. Die Umstellung kompletter Systeme auf diese Bipolar-Logik-ICs wirft das Problem der geringen Geschwindigkeit in der Umgebung der ICs auf.
Unterhalb der 0,7-μm-Grenze werden Gate-Verzögerungen von 200 bis 600 ps, bei Bipolar-ICs 80 bis 200 ps erwartet.
Die Grenze von 0,25 μm wird in den nächsten zwei Jahren kaum unterschritten werden können. Erst Mitte der 90er Jahre soll diese Technologie im Labor beherrschbar sein.
Die Ankündigung japanischer Firmen, bereits 16-MBit-DRAMs produzieren zu können, wird mit Skepsis aufgenommen. Die Realisierung von Chips mit Linienbreiten unter 0,7 μm setzt die Beherrschung völlig neuer Bearbeitungstechniken voraus. Die wahrscheinlich erfolgreichste Technologie wird die Röntgenstrahl-Lithographie sein. Sie hat ihre Eignung bereits nachgewiesen, ist jedoch aufgrund der geringen Durchsatzgeschwindigkeit noch nicht für die Massenproduktion geeignet. Der Einsatz anderer Materialien (z. B. GaAs) macht sich noch nicht zwingend erforderlich, da die bipolare Silizium-Technologie noch nicht ausgereizt ist.
Ein anderer Weg, den Integrationsgrad zu erhöhen, ist der Übergang zu 3D-Chips. Einige Firmen, darunter IBM, arbeiten an diesen Chips, bei denen mit konventionell beherrschbarer Linienauflösung gearbeitet wird, allerdings mit mehreren aktiven Schichten. In diesem Jahr wird mit den ersten 3D-Chips mit 2 Systemlagen gerechnet. Die obersten Lagen dieser ICs sollen zur Geschwindigkeitssteigerung mit optoelektronischen Elementen versehen werden.

Quelle: eee. Elektronik-Technologie. – Leinfelden-Echterdingen (1989)1/2. – S. 19, 20 *Wi*

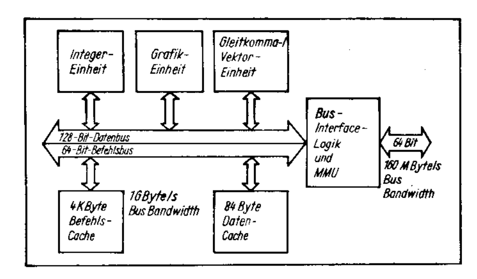
Optisches Netz mit Kunststoffasern

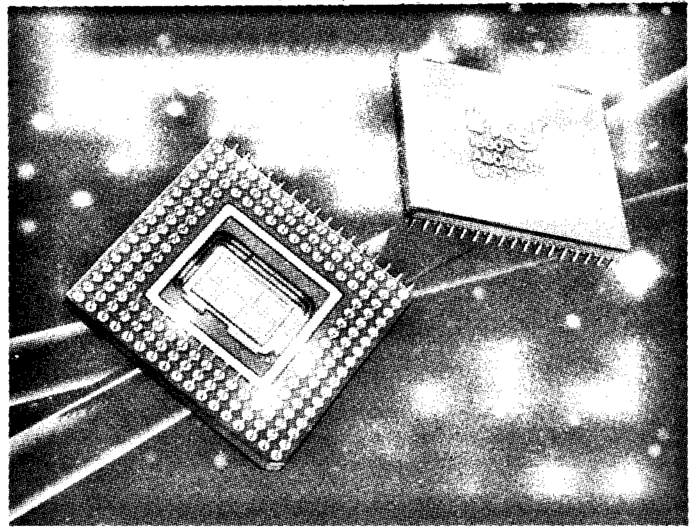
Ein optisches LAN aus Kunststoffasern mit der Bezeichnung *Fiberstar* und mit einer Übertragungsrate von 2 MBaud wurde kürzlich von der USA-Firma Netronix vorgestellt. Das Netzwerk mit Kabellängen von 15 m und einer theoretischen Übertragungsrate (bei Einsatz von Laserdioden statt Leuchtdioden als Sender) von 10 bis 50 MBaud soll das erste Kunststoffasernetz sein. Vorteile der Kunststoffasern gegenüber Glasfasern sind der geringere Preis von Fasern und Steckverbindern, die bessere elektromagnetische Verträglichkeit, die bessere Flexibilität und der kleinere mögliche Biegeradius. Weiterhin kann wegen der Arbeit im Bereich des sichtbaren Lichtes die Fehlersuche vereinfacht werden. Die verwendete Kunststoffaser ist mit einem Durchmesser von 1 mm erheblich dicker als Glasfasern (62 μm). *MP*

vorgestellt

Intel i860

Wegbereiter einer neuen Leistungsklasse bei Mikroprozessoren





Im Rahmen der ISSCC '89 in New York stellte Intel einen neuen Mikroprozessor vor: den i860 (zunächst auch unter dem Namen N10 oder ONNY – „officially not named yet" – bekannt). In fünf Jahren Entwicklungszeit wurde im Rahmen eines eine Milliarde US-$ teuren Projekts (davon wurde ein Drittel für eine neue Fertigungsstätte aufgewendet) ein neuer Hochleistungsmikroprozessor entwickelt:

- gemischte 32-/64-Bit-Architektur
- drei interne, parallel arbeitende Ausführungseinheiten
- Verarbeitungsleistung maximal 120 Millionen Operationen je Sekunde, 33 VAX-MIPS, 10 MFLOPS, 500000 Grafiktransformationen je Sekunde (alle Angaben für 40-MHz-Taktfrequenz)
- 1 μm CHMOS-IV-Technologie, mehr als 1 Million Transistoren, Chipgröße 1,5 cm mal 1 cm
- Taktfrequenz zunächst 33 MHz, künftig 40 und 50 MHz
- Preis 750 US-$ (ab 1000 Stück).

Worin besteht der konzeptionelle Unterschied zwischen dem Intel 80860 (i860) und dem Intel 80486 (i486)? Der i486 ist eine Reimplementierung des i386 in einem höheren Technologieniveau. Dabei kann ein funktionell erweiterter 32-Bit-Prozessor i386 gemeinsam mit einem i387-Subset sowie Daten- und Programmcaches auf einem Chip integriert werden. Somit besteht Softwarekompatibilität in der iX86-Familie. Dank der leistungsfähigeren Technologie wird die Befehlsausführung auf dem i 486 gegenüber dem i386 um den Faktor 2 bis 3 beschleunigt. (Siehe auch MP 12/1988, Seite 384.) Der i860 hingegen zielt auf ein neues Anwendungsgebiet für Mikroprozessoren und weist dementsprechend neue Architekturmerkmale auf: Grafiksuperworkstations und Superrechner. Er verfügt über drei separate, parallel arbeitende Ausführungseinheiten.

Die 32-Bit-Integereinheit ist nach dem RISC-Konzept gestaltet: gezielte Einschränkung der Vielfalt an Datentypen und Befehlen zugunsten einer schnellen Befehlsausführung und eines niedrigen Anteils an Chipfläche. Für 8-, 16-, 32- oder 64-Bit-Daten sind etwa 40 Basisbefehle definiert: Speicherzugriff (Lesen und Schreiben), Registertransfer, Addition und Subtraktion, Schiebebefehle, Logikbefehle, Programm- und Systemsteuerbefehle. 32 Datenregister zu je 32 Bit sowie einige Steuerregister sind diesem RISC-Kern zugeordnet. Die Gleitkommaeinheit kann wahlweise Daten mit 32 und 64 Bit verarbeiten (Erfüllung der Forderungen des Standards ANSI/IEEE 754-1985). Sie verfügt über getrennte Multiplizier- und Addiereinheiten. Daten können in einem 32 × 32-Bit-Registerfile mit fünf Toren zu verschiedenen internen Bussen in einer Organisation zu 32, 64 oder 128 Bit (Long-Integer) abgelegt werden. Unterstützt werden die Grundrechenarten, Quadratwurzel, Reziprokwert, Zahlenvergleiche sowie Befehle für Long-Integerdaten. Diese Befehle sind in verschiedenen Varianten zur optimalen Unterstützung der Parallelarbeit im Prozessor verfügbar. Eine besondere Gruppe von Doppelbefehlen ermöglicht in einem Befehl eine Kombination aus Addition bzw. Subtraktion und Multiplikation. Solche Befehle werden für die Vektorverarbeitung und für die Signal- und Bildverarbeitung (Faltungsoperation) benötigt.

Die dritte Einheit verarbeitet 3D-Grafikdaten (8, 16 oder 32 Bit). Wahlweise können diese Daten Intensitäts- und Farbinformationen in verschiedenen Auflösungsstufen sowie weitere Attribute und Texturen enthalten. Die Grafikbefehle beziehen sich auf einen Z-Puffer, in dem die dritte Koordinate eines räumlich lokalisierten Bildpunktes ausgewertet wird. Bilddaten können automatisch interpoliert und skaliert werden. Die Pixelspeicherung im Bildspeicher ist selektiv möglich (Unterdrückung von verdeckten Kanten).

Interne 4-KByte-Befehls- und 8-KByte-Datencaches sowie die Registerfiles gewährleisten, daß Daten und Befehle für Prozeduren (lokaler Programmkontext) auf dem Chip gehalten werden können und so ohne Speicherzugriff schneller erreichbar sind. Zwischen den Funktionseinheiten des i860 werden Daten über einen 128-Bit-Bus mit bis zu 533 MByte je Sekunde und Befehle über einen 64-Bit-Bus mit bis zu 266 MByte je Sekunde übertragen. Zusätzlich gibt es weitere Koppelbusse zwischen den einzelnen Einheiten, so daß die On-Chip-Datentransferrate insgesamt etwa 1,2 GByte je Sekunde beträgt. Das Businterface des i860 gestattet die parallele Verarbeitung von drei Buszyklen. Die Konzepte der virtuellen Speicherorganisation und der Cacheverwaltung wurden zum Teil dem i386 entlehnt: Adreßraum 4 GByte, seitenorganisierter Hauptspeicher (optimiert für DRAMS ab 256 KBit), *Translation Lookaside Buffer* mit 64 Referenzen zu Speichersegmenten in zwei Privilegierungsmodi, Speicherbankadressierung über 8 Enablesignale.

Die Interruptreaktionszeit beträgt maximal 66 Taktzyklen. Für Multiprozessorsysteme gibt es einen Busverriegelungsbefehl zur Gewährleistung eines ungeteilten Lese-/Schreibzugriffs auf den Speicher (LOCK) sowie ein Busrequestsignal für einen externen Busarbiter.

In der Prozessorarchitektur wurden weitere geschwindigkeitserhöhende Maßnahmen vorgesehen: Pipelineorganisation der Verarbeitungs- und Buszugriffseinheiten, Blocktransfer zur Cacheaktualisierung, überlappende Buszyklen (Aussenden der nächsten Adressen bereits während des Lesens der Daten des vorherigen Buszyklus), Unterstützung von Static-Column- oder Page-Mode-DRAMs.

Diese Architekturkonzepte konnten nur mit den Mitteln der Höchstintegration und unter Anwendung des RISC-Konzepts in einem Chip umgesetzt werden. Der i860 ist praktisch ein Multiprozessorsystem aus Spezialprozessoren auf einem Chip. Mit den eingangs genannten Verarbeitungsleistungen zielt der i860 auf einen neuen Anwendungsbereich für Mikroprozessoren: Supergrafikworkstations und Supercomputer. Nachdem Intel bereits Industriestandards für Mikroprozessoren der PC-Klasse (i286) und der Workstationklasse (i386) setzte, wird nun dieser Spitzenanwendungsbereich der modernen Rechentechnik erschlossen, für den in den kommenden Jahren besonders hohe Wachstumsraten erwartet werden. Bisher mußten Hochgeschwindigkeitstechnologien (ECL und GaAs) und/oder Multiprozessorsysteme (z. B. iPSC286-Hypercube) angewendet werden, um Superrechenleistung zu erreichen. Solche Lösungen sind aber sehr teuer und für die Masse der potentiellen Anwender unökonomisch. Supermini- und Superrechner kosteten bislang mehrere 100000 bis Millionen US-$. Für nur etwa 10000 US-$ kann nun ein IBM-AT um eine i860-Ergänzungskarte erweitert werden (Acceleratorboard). Superrechenleistung kommt damit auf den Schreibtisch des Anwenders, der künftig in einer UNIX-V.4.0-Umgebung in Fortran und C vor allem folgende Problemklassen effektiv bearbeiten kann: CAD – beispielsweise VLSI-/ULSI-Schaltkreisentwurf, bewegte 3D-Simulations- und Grafikprobleme („4D-Grafik", z. B. Simulation von Strömungsprozessen), Klimaforschung/Meteorologie (mittelfristige Wetterprognose) oder Bildverarbeitung. Solche Aufgaben können in skalare und vektorielle Verarbeitungsanteile, die sich parallel abarbeiten lassen, zerlegt werden. Der i860 verfügt dafür über eine adäquate Prozessorarchitektur. Bei Vektorisierungsgraden von 50 bis 80% erreicht er noch mindestens 75% der Leistung einer CRAY1-80, und das zu einem Bruchteil des Aufwands. Vor allem der Schaltkreisentwurf erfordert zunehmend leistungsfähigere Rechner. Intel setzt bereits ein iPSC-Hypercube-System für Simulationszwecke ein. AT & T mußte beim Entwurf des RISC-Prozessors CRISP (Integrationsgrad 172000 Transistoren) auf ein ALLIANT-Minisuperrechnersystem zurückgreifen. Mit der zunehmenden Verbreitung höchstintegrierter anwendungsspezifischer Schaltkreise (ASICs) wird der Bedarf an hochleistungsfähiger, aber billiger Rechentechnik wachsen. Supercomputer und Superworkstations sind als Stimulus und zugleich als Randbedingungen für den Fortschritt beim VLSI-/ULSI-Schaltkreisentwurf anzusehen.

Etwa 1991/92 ist mit einem Multiprozessorsystem auf der Basis des i860 aus bis zu 1000 Prozessoren zu rechnen. Damit werden Verarbeitungsleistungen von etwa 20 Milliarden Gleitkommaoperationen je Sekunde (GFLOPS) erreichbar sein (Supercomputer des oberen Leistungsbereichs). Ein entscheidendes Problem, das bis dahin aber gelöst werden muß, ist die Bereitstellung geeigneter Softwareentwurfsumgebungen für die Anwender: Nur wenn eine effektive Zerlegung der Algorithmen in parallel abzuarbeitende Teilaufgaben gelingt, kann das Leistungsvermögen des Parallelrechners ausgeschöpft werden. Dazu bedarf es neuer, anwenderfreundlicher Programmierwerkzeuge.

guv

MODULA-2

von H. Schiemangk, 1. Aufl., VEB Verlag Technik, Berlin 1989, 240 S., 4 Bilder, 6 Taf., DDR 24,– M

MODULA-2 ist eine kleine, überschaubare Sprache, die zu diszipliniertem Programmieren erzieht. Für alle gängigen 16- und 32-Bit-Rechner existieren inzwischen leistungsfähige MODULA-2-Compiler und dazugehörige Programmierumgebungen. MODULA-2 unterstützt den Aufbau komplexer Programme aus kleinen übersichtlichen Bausteinen, genannt Module, die separat compiliert und getestet werden können. Die Schnittstellen zwischen den einzelnen Bausteinen werden dabei vom Programmierer separat spezifiziert, und der Compiler überwacht dann die Einhaltung dieser vordefinierten Schnittstellen.
Mit dem Buch setzt sich der Autor das Ziel, Programmieranfänger an die Sprache MODULA-2 heranzuführen. In einem einleitenden Abschnitt wird ausführlichst an Hand eines leicht verständlichen Beispiels die Arbeitsweise mit Bausteinen in MODULA-2 erläutert. Es folgt ein Abschnitt, in dem die Arbeitsweise eines MODULA-2-Systems aus der Sicht des Anwenders beschrieben wird, das heißt der Weg vom Quelltext zum abarbeitbaren Programm. Die benutzten Kommandos entsprechen dabei im wesentlichen dem MODULA-2-System des Kombinates Robotron unter DCP.
Den Hauptteil des Buches bildet die Sprachbeschreibung von MODULA-2. Dabei sind in jedem Kapitel ausführliche nichttriviale Beispiele enthalten, die dem Lesenden sowohl das Verstehen erleichtern als auch Denkanstöße geben und Programmiertechniken vermitteln. Der Autor stellt alle wesentlichen Sprachelemente und Konstruktionen der Sprache MODULA-2 ausführlich und verständlich dar. Sehr kurz werden lokale Module und die Möglichkeiten der Benutzung des Moduls „SYSTEM" dargestellt. Nicht behandelt wird das Koroutinenkonzept von MODULA-2. Die in den meisten MODULA-2-Systemen vorhandenen Möglichkeiten der maschinennahen Programmierung werden ebenfalls nicht berücksichtigt.
Abschließend kann festgestellt werden, daß dieses Buch für jeden Programmieranfänger bzw. jeden Programmierer, der sich mit MODULA-2 vertraut machen will, empfehlenswert ist. Es ist ein didaktisch gut aufgebautes Lehrbuch für MODULA-2. Als Nachschlagewerk für den versierten MODULA-2 Programmierer ist es nur bedingt geeignet.

Dr. J.-P. Bell

TECHNIK EN MINIATURE

Stippvisite bei der Mikrotechnik

von B. Winde und J. Heim, 1. Auflage, Urania-Verlag Leipzig/Jena/Berlin, 1988, 120 S., 34 Abbildungen, DDR 7,50 M

Ein unkomplizierter Titel für eine komplizierte Sache. Bertram Winde und Joachim Heim versuchen, sich dem Begriff Mikro von verschiedenen Seiten zu nähern. Was sie dabei erreichen wollen? Einerseits am Beispiel von Makrotechniken zu zeigen, wie Mikrotechniken funktionieren bzw. wie diese sich aus Makrotechniken entwickeln und andererseits den engen Zusammenhang verschiedener Mikrotechniken aufzuzeigen: „Die Mikroelektronik als technologische Wiege für Mikromechanik, Mikrooptik und weitere Mikrotechniken".
Das Lesen dieser Broschüre im Westentaschenformat fällt leicht. Sie ist mit vielen interessanten Vergleichen versehen, die ständig zum Weiterlesen anregen. Hinzu kommen ein guter Sprachstil und eine sehr gute didaktische Aufarbeitung. Das Büchlein eignet sich deshalb für jeden, der wissen möchte, welche Möglichkeiten die Mikroskopie (eine Voraussetzung für jede Mikrotechnik) bietet, was Tomographie ist, wie sich die Computer entwickelten und wie diese sich in ihrer Leistung vom menschlichen Gehirn unterscheiden. Nicht zuletzt wird „fast nebenbei" erklärt, wie ein integrierter Schaltkreis hergestellt wird. Auch die Optoelektronik wird gestreift: vom Bildleitkabel über Lichtwellenleiter bis hin zu einem der neuesten Gebiete der Halbleitertechnik, der integrierten Optik. Das wohl Faszinierendste ist jedoch die Mikromechanik. Ausgehend von der Funktionsweise unseres Ohres beschreiben Bertram Winde und Joachim Heim die Herstellung von einem Mikrofon und von einem kompletten Gaschromatographen auf einer Siliziumscheibe. Die rasante Entwicklung der Mikrotechniken einschließlich der Mikromechanik lassen das Stellen von Prognosen immer schwieriger werden. So ist beispielsweise die Aussage, daß die Miniaturisierung von Drehteilen oberhalb von 100 μm aufhört, schon wieder überholt; neueste Meldungen aus der Universität Berkeley berichten über die Entwicklung eines Motors mit einem Durchmesser von nur 60 μm.
Mit der Art und Weise der Darstellung ist die vorgestellte Broschüre geeignet, einem sehr breiten Leserkreis Begriffe, die oft nur als Schlagwörter bekannt sind, anschaulich und leicht verständlich zu erklären. Die Autoren, von Beruf Hochschullehrer, setzen damit Maßstäbe. Sie werten die Technik en miniature zwar als nur *einen* Aspekt der Entwicklung von Schlüsseltechnologien, aber als einen sehr wichigen: „Denn auch in Zukunft gilt eine jahrhundertealte Spruchweisheit: Klein ist fein".

H. Hemke

dBase III Plus in 100 Beispielen

von R. Dierenbach und W. Mehl, B. G. Teubner Stuttgart: 1989, 300 S., DM 32,–, ISBN 3-519-02546-9

Das gegenwärtig für IBM-kompatible PCs wohl am weitesten verbreitete Datenbankbetriebssystem dBase III Plus wird mit dem vorliegenden Buch auf äußerst angenehme, ja unterhaltsame Weise dargeboten. Dazu trägt auch die ansprechende Gestaltung bei. In den 13 Abschnitten kann sich jeder Computerneuling von den Anfängen, der Funktion eines PC, über den Aufbau und die Nutzung von Datenbanken und deren Auswertung bis hin zur Anwendung von "dBase III Plus im Netzwerk" verführen lassen.
Die wichtigsten Befehle zum Anlegen einer dBase III Plus-Datenbank sind in einem speziellen Abschnitt zusammengefaßt und werden an Beispielen ausführlich erläutert.
Die dabei dem ernsthaften Thema zunächst unangemessen erscheinende Auswahl des Beispiel-Datenbankinhalts lockert jedoch meines Erachtens den Umgang mit einer gerade für den Laien schier unübersehbaren Fülle von gleichzeitig einstürmenden Informationen erheblich auf und läßt den Umgang mit dem PC geradezu spielerisch werden.
Daß der Wissenschaftlichkeit dadurch kein Abbruch getan wird, zeigt die exakt beschriebene Syntax der einzelnen Befehle – auch hier natürlich eine leicht verständliche Erläuterung kompliziertester Vorgänge. Wer sich trotzdem noch unsicher wähnt, kann sich im gleichnamigen Abschnitt über den Assistenten als dem „Stützrad von dBase III Plus" ausgiebige Hinweise holen.
Neben den Funktionen von dBase III Plus werden die Befehle zum Sortieren und Indizieren sowie die integrierten Generatoren für Listen und Aufkleber ausführlich erläutert.
Breiten Raum nimmt auch der Abschnitt „Programmieren mit dBase III Plus" ein. Können doch erst durch diese Kenntnisse die Möglichkeiten sowohl des Einzelnutzerbetriebes als auch der abschließend beschriebenen Netzwerknutzung voll ausgeschöpft werden. Für den (potentiellen) Programmierer sind darüber hinaus einige Tips und Tricks im Umgang mit dBase III Plus enthalten. Die beiden Beispielprogramme zeigen die wichtigsten Standardsituationen (Editieren, Sortieren und Anzeigen von Daten, Menüs und anderes sind auf unterschiedliche Weise programmiert).
Nützlich für viele dBase II-Nutzer wären zum Übergang zu dBase III Plus meines Erachtens einige Hinweise, wie die mühsam eingetippten Namen, Telefonnummern oder ähnliches nun mit einem neuen PC weiterverwendet werden können. Dierenbach und Mehl hüllen sich leider in Schweigen über dieses ihnen sicher bekannte Thema. Vom Verlag wünsche ich mir für dBase IV eine noch besser auf teilweise verwirrende Druckfehler durchgesehene Ausgabe.
Alles in allem liegt uns ein Buch vor, das dem Einsteiger ein systematisch aufgebautes Lehrmaterial und dem Fortgeschrittenen ein angenehmes Nachschlagewerk sein dürfte.

H.-J. Hill

Künstliche Intelligenz

KI – Einführung und Anwendungen

von E. Rich, McGraw-Hill Book Company GmbH, Hamburg 1988, 468 S., etwa 180 Literaturstellen

Elaine Rich, Pionier der Forschung und Lehre zur künstlichen Intelligenz in den USA, legt ein Buch vor, das mit der KI selbst gewachsen und damit zeitlos geworden ist. Das Buch gibt eine sehr gute Einführung in die Begriffswelt der KI. Es bietet alle Voraussetzungen für ein autodidaktisch erwerbbares, tiefes Verständnis der Grundbegriffe und ihrer Beziehung zueinander. Es ist Einführung und Nachschlagewerk zugleich.
Lehrbuchhaft verbindet es Stoffvermittlung mit Stoffvertiefung und Anwendung durch zahlreiche anschauliche Übungen. Das erste Kapitel erörtert die Frage: „Was ist künstliche Intelligenz?" wobei die „Väter der KI" Newell, Simon, Feigenbauen und Robert Rich insbesondere Pate standen. Das zweite Kapitel behandelt das KI-Zentralproblem unserer Zeit, das *Problemlösen* als eine zielgerichtete, selbständige Bewegung durch einen Wissensraum. Folgerichtig beschreibt das vierte Kapitel eine Auswahl grundlegender Problemlösungsverfahren als Operationen in Zustands-Handlungs-Bäumen. Dazu wird die Wissensbasis mit geeigneten Merkmalen als Problemlösungsobjekt eingeführt.
Besonders hervorgehoben sind die „schwachen" *Suchverfahren* im Unterschied zu den „direkten": Erzeugen und Prüfen; Bergsteigen; Breitensuche; Bestensuche u. a. Eine entsprechende Bewertung erhalten geeignete Suchalgorithmen. Spiele sind sehr anschauliche Beispiele im 4. Kapitel.
Der zweite Teil des Buches (Kapitel 5, 6, 7) befaßt sich sehr systematisch mit der *Wissensdarstellung*, wobei die *Prädikatenlogik* als Grundverfahren der Wissenspräsentation erscheint. Ein Schlüsselbegriff ist die „natürliche *Deduktion*" als wichtigstes Schließverfahren. Ergänzend werden die „nichtmonotone Logik" und die „Wahrscheinlichkeitsschlüsse" behandelt, wobei die praktische Bedeutung jeweils hervorgehoben wird.
Als ein wichtiger – vielleicht entscheidender Aspekt der Wissensdarstellung – wird die *Strukturierung* besonders aus der Sicht der deklarativen Repräsentation beschrieben: semantische Netze, Rahmen, Spripte.
Unter dem Aspekt „Fortgeschrittene Themen" baut der dritte Teil (Kapitel 8, 9) die Grundlagenthemen aus, um zugleich z. B. mit Planungstechniken (einfache, nichtlineare, hierarchische) das Begriffsfeld mit Lösungsverfahren auf Expertensystemen zu erweitern.
Nahezu außerhalb der inneren Logik des Buches – aber ganz wichtig für die Mensch-Maschine-Schnittstelle – wird die Verarbeitung der *natürlichen Sprache* behandelt (Gestaltung der Dialog- und Erklärungskomponente).
Das Buch ist in einer didaktisch guten Sprache geschrieben. Es ist nahezu voraussetzungsfrei (Ingenieurkenntnisse) lesbar und somit ein „immer gültiges Einstiegswerk" mit gehobenem Anspruch. Da Programme nicht enthalten sind, gibt es auch kein Problem ihrer Implementierbarkeit auf verfügbaren Maschinen. Wer sich mit KI beschäftigen will, sollte dieses Buch lesen. *Prof. Dr. Dr. M. Roth*

Leipziger Frühjahrsmesse 1989

Der erste Teil unseres Berichtes von der Leipziger Frühjahrsmesse 1989 in MP 7/1989 widmete sich dem Leitthema der Messe „Flexible Automatisierung" sowie den Rechnern der Klassen Mini und Mikro. Im zweiten Teil wollen wir ihnen Exponate der Computerperipherie, von Applikationen und Netzen von PCs, von Bauelementen sowie Software vorstellen.

Peripherie

Eine unabdingbare Voraussetzung für Computer Aided Design (CAD) ist die Möglichkeit der Zeichnungsherstellung. Dazu werden meist Plotter eingesetzt. Der neue A3-Plotter **K 6416** vom Kombinat Robotron (Bild 1; Farbbilder siehe 2. und 3. Umschlagseite) hat eine Auflösung von 0,01 mm und eine Zeichengeschwindigkeit von 300 mm/s. Durch erweiterten Randmodus können technische Zeichnungen mit standardgerechter Größe der Umrandungen und des Schriftfeldes erzeugt werden. Es können sowohl weißes Papier als auch Transparentpapier oder Folien in 8 Strichstärken oder Farben beschriftet werden. Der Datenpuffer hat eine Größe von 14 KByte.

Das polnische Unternehmen Z. M. P. Mera Poltik Lodz bot den A4-Plotter **MDG-1** mit einer mittleren Zeichengeschwindigkeit von 63 mm/s und einer Auflösung von 0,2 mm an. Mit seiner kompakten Bauweise ($300 \times 190 \times 85\,mm^3$) eignet er sich sehr gut für die Aufstellung in Büros. Als Schnittstellen stehen Centronics und V.24 zur Verfügung. Er besitzt 4 Stifte und kann in den zwei Betriebsarten Text und Grafik arbeiten.

Als Drucker verwendbar ist die ebenfalls auf der LFM '89 erstmals ausgestellte Schreibmaschine **Erika electronic S 6007** von Robotron (Bild 2). Gegenüber dem Vorgängermodell Erika S 6006 hat sie den Vorteil, daß sich das Typenrad zum schnelleren Wechsel in einer Kassette befindet; auch der Farbbandwechsel wurde wesentlich vereinfacht. Statt der bisher 3 Schnittstellen-Module besitzt die S 6007 nur *ein* Uni-Modul (im Bild 2 links), das die Schnittstellen Centronics, Commodore und V.24 enthält. Typenräder mit den Zeichensätzen IBM, Schneider, Commodore und Schreibmaschine sind verfügbar. Die Arbeitsweise der einzelnen Schnittstellen kann mit den im Uni-Modul vorhandenen DIL-Schaltern modifiziert werden.

Die Firma Epson hatte neben Personal- und Handheldcomputern selbstverständlich auch ein Sortiment an Drucktechnik ausgestellt. Darunter den ersten 48-Nadeldrucker der Welt, den **TLQ 4800**, für den eine Auszeichnung für gutes Design in Empfang genommen werden konnte. Auf eine Beschreibung des Druckers sei hier verzichtet, da wir ihn bereits in MP 6/88 ausführlich vorstellten. Weitere Spitzenprodukte waren der Tintenstrahldrucker SQ 2550 und der neue Laserdrucker GQ 5000.

Bei dem **SQ 2550** (Bild 3) wird besonders hervorgehoben, daß nun auch bei einem Tintenstrahldrucker der hohe Bedienkomfort geboten wird, den der Anwender von Nadel-Matrixdruckern kennt. Der SQ 2550 kann sowohl Endlospapier als auch Einzelblätter bis zum Format A3 verarbeiten (das A4-Modell heißt SQ 850). Eine neue Parkposition ermöglicht die parallele Verarbeitung der verschiedenen Papierarten ohne umständliche Umbauten am Drucker. Das Heranfahren des Endlospapiers an die Abreißkante nach der Abarbeitung der anliegenden Daten sowie der Einzug von Einzelblättern über wahlweise ein oder zwei Schächte werden automatisch ausgeführt. Der halbautomatische Einzelblatteinzug über Schubtraktor und Friktionswalze ist Standard, optional ist ein Aufsatz-Zugtraktor. Der Druckkopf besitzt 24 Düsen und eine Reinigungsautomatik für einen wartungsfreien Betrieb. Damit wird die hohe Grafikauflösung von 360 dpi (dots per inch – Punkte/Zoll) ermöglicht, so daß der Drucker auch für den Einsatz im CAD- und im DTP-Bereich gut geeignet ist. Die Druckgeschwindigkeit beträgt bei Schnellschrift bis zu 594 Zeichen/s und bei Schönschrift bis 198 Zeichen/s. Auf Durchschläge muß man beim Tintenstrahldruck jedoch bekanntlich ebenso wie beim Laserdrucker verzichten.

Dies trifft also auch auf den Laserdrucker **GQ 5000** zu (Bild 4), mit dem Epson nun auch dieses Marktsegment in seiner Produktpalette ausfüllt. Der GQ 5000 ist ein typischer Vertreter der gegenwärtigen leistungsfähigen Tisch-Laserdrucker. Er liefert wie die meisten Geräte dieser Klasse eine Auflösung von 300 Punkten/Zoll und hat eine Druckgeschwindigkeit von 6 Seiten/Minute. Eine Neuheit soll dagegen das zweizeilige Bedienfeld darstellen, über das man sämtliche Funktionen des Druckers speichern und aufrufen kann und das den Umgang mit dem Gerät bedeutend erleichtern soll. Hervorgehoben wird auch die Möglichkeit, in drei Slots gleichzeitig IC-Karten mit verschiedenen Zeichensätzen laden zu können.

Auch der – wie Epson zur japanischen Seiko Group gehörende – Druckerhersteller Seikosha hatte zur LFM '89 seine Spitzenprodukte mitgebracht. So den schnellsten Matrixdrucker der Welt, den **SBP-10 AI** (vorgestellt in MP 2/89, S. 60) und den neuen 24-Nadeldrucker mit Programmkarte, den **SL-230 AI** (vorgestellt in MP 7/89, 3. Umschlagseite). Neu ist ebenfalls der 9-Nadeldrucker **SP-1600 AI** (Bild 5). Er hat eine Druckgeschwindigkeit von 160 Zeichen/s bei Schnellschrift und 27 Zeichen/s bei NLQ. Die 11 internationalen Zeichensätze können über Software und DIP-Schalter angewählt werden. Der 2,3 KByte große Druckpuffer läßt sich auf bis zu 8 KByte erweitern. Unser Bild zeigt den SP-1600 AI und die automatische Einzelblattzuführung, mit welcher der Drucker anstelle der serienmäßigen halbautomatischen Einzelblattzuführung ausgestattet werden kann.

Als einer der bekanntesten Produzenten von Festplattenlaufwerken gilt die US-amerikanische Seagate Technology Inc. Die Firma ist seit 1979 auf dem Markt und soll – nach eigenen Angaben – mit einem Ausstoß von 20 000 Einheiten pro Tag gegenwärtig zwei Drittel der in der Welt produzierten PCs mit Festplattenlaufwerken ausstatten. Aus dem reichhaltigen Angebot wurden in Leipzig die beiden 5,25-Zoll-Winchestermodelle ST 225 N und ST 238R vorgestellt (Bild 6). Das **ST 225N** hat eine formatierte Kapazität von 21,3 MByte und eine mittlere Zugriffszeit von 20 ms. Die Aufzeichnungsdichte beträgt 9,83 bpi (bit per inch) bei dem MFM-Aufzeichnungsverfahren. Als Interface dient das schnelle SCSI. Das Laufwerk **ST 238R** hat eine formatierte Kapazität von 32,7 MByte (unformatiert 38,4 MByte) und ebenfalls eine mittlere Zugriffszeit von 20 ms. Die Aufzeichnungsdichte beträgt hier dank des RLL-(2,7)-Aufzeichnungsverfahrens 14,7 bpi. Es wird das Interface ST 412 verwendet.

Die Firma RISTO ist seit Jahren auf der Leipziger Messe als Aussteller von Computermöbeln und nützlichem Zubehör vertreten. In diesem Jahr zeigte man als Komplettierung eines Computerarbeitsplatzes den neuen Kleinschneider **Böwe 355**, der speziell auf die Belange des Büroalltags abgestimmt ist (Bild 7). Vom dünnen Papier ($40\,g/m^2$) bis zum $200\,g/m^2$ schweren Karton werden Vordrucke rundum exakt geschnitten, und zwar auch Mehrfachsätze. Die Bedienung ist ähnlich einfach wie bei Kopierern: 16 verschiedene Formate werden entweder schon bei Lieferung des Gerätes eingegeben oder später vom Bediener selbst abgespeichert und bei Bedarf über das Bedienpaneel abgerufen.

Im Zusammenhang mit der Vorstellung von Modellen des neuen IBM PS/2 zur LFM '88 hatten wir auf das Problem verwiesen, welches damit entstand, daß IBM 3,5-Zoll-Disketten als neuen Standard wählte. Das heißt, die neuen Modelle können mit den standardmäßigen Laufwerken nicht mehr die 5,25-Zoll-Disketten verarbeiten, die bei PC-Nutzern in den letzten Jahren eine große Verbreitung gefunden haben. Um dem abzuhelfen, boten schon bald Hersteller externe Laufwerke an, und seit einiger Zeit gehört auch IBM dazu. Das Bild 8 zeigt ein solches **Beistellgefäß** am PS/2-Modell 30 im Angebot der Firma IBM ROECE aus Wien.

Applikationen und Netze mit PCs

Für das Publizieren am Schreibtisch (DTP – desktop publishing) hat die Firma Apple um die Jahreswende einen neuen Anwendungsschwerpunkt propagiert. Anhand von Untersuchungen soll festgestellt worden sein, daß der typische Ingenieur nur etwa ein Drittel seiner täglichen Arbeitszeit mit dem Konstruieren oder anderen technisch-wissenschaftlichen Aufgaben verbringt. Der weitaus größere Teil dagegen entfällt auf das Abfassen von Konzepten, Kalkulationen, Berichten oder das Erstellen von Dokumentationen. Hier sieht die Firma den Ansatzpunkt für das „Desktop Engineering" und damit verbunden natürlich wachsende Absatzchancen. Profitieren von einer solchen Argumentation dürften dabei sicher auch alle anderen Anbieter von DTP-Systemen.

Die Handelsfirma Transcommerz bot zur LFM eine **DTP**-Konfiguration an, deren Basis ein 286er PC war, der, wie auch ein XT-kompatibler PC, unter dem Namen Büro Datic von der Firma seit kurzem selbst gefertigt wird (Bild 9). Als Layoutprogramm zum Gestalten von einfachen Formularen, Prospekten und Dokumenten unter Einbeziehen von Bildern und Grafiken dient das DTP-Programm PageMaker. Das Eingeben der Dokumente, Grafiken und Fotos erfolgt beispielsweise über den neuen Tisch-Scanner HP ScanJet PLUS mit einer Auflösung von 300 Punkten pro Zoll. Von seinem Vorgänger unterscheidet sich der „PLUS" durch die Möglichkeit von jetzt wahlweise 64 oder 256 Graustufen, 256 Kontraststufen, 256 Helligkeitsstufen und die höhere Scangeschwindigkeit von 10 Sekunden für eine A4-Seite (bei 300 dpi). Als Ausgabemedium war der Tischlaserdrucker Laserstar 6 von AEG Olympia angeschlossen. Das Gerät leistet eine Druckgeschwindigkeit von 6 Seiten/Minute und hat eine Auflösung von 300 Punkten/Zoll.

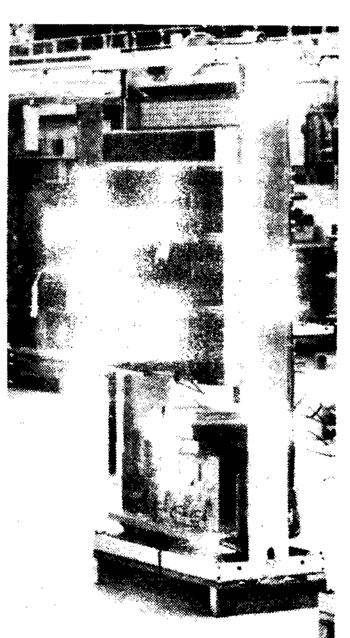
Mit der Entwicklung der Computertechnik wächst auch die Nachfrage nach der Kopplung der Rechner. Dabei steht weniger das Übermitteln von Informationen als das gemeinsame Nutzen großer Datenbestände im Vordergrund. Die zur Zeit noch am häufigsten eingesetzten Rechnernetze sind die **LANs** (Local Area Network). Das Kombinat Robotron war dieses Mal mit dem neuen **LAN EC-NET** vertreten. Es ist aus ROLANET 1-Baugruppen aufgebaut und dient ausschließlich dem Vernetzen der Personalcomputer EC 1834. Das Programmpaket EC-NET unter dem Betriebssystem DCP 3.30 bedient die international verbreitete NETBIOS-Schnittstelle. Damit sind solche Funktionen wie Senden, Empfangen und Speichern von Nachrichten sowie das Bereitstellen von Disketten, Verzeichnissen und Druckern zur gemeinsamen Nutzung im LAN verfügbar. Der Zugriff zum Netz ist über Programme wie dBase III Plus und ARIADNE, über Programmiersprachen wie C und T-PASCAL sowie über DCP-Kommandos wie DIR und COPY möglich.

Weiterhin wurden von Robotron neue Anwendungen für das **SCOM-LAN** vorgestellt. So wurde neben Programmen für Literaturrecherche und gemeinsame Datenbanknutzung die Kopplung mit digitaler Richtfunktech-

nik (Bild 10) demonstriert. Zum Einsatz kommen die digitalen Richtfunkeinrichtungen PCM 10-300/400/800. Sie dienen der Datenübertragung über große Entfernungen und bei schwierigen Geländeverhältnissen. Dabei können sowohl einzelne entfernte SCOM-LAN-Teilnehmer als auch mehrere LANs gekoppelt werden. Der Abstand zwischen zwei Richtfunkstationen kann 50 km betragen. Ein Übertragungsweg enthält 10 Kanäle mit einer Übertragungsgeschwindigkeit von bis zu 64 kBit pro Sekunde. Ist die Anzahl der anzuschließenden Datenstationen größer als 10, werden Datenmultiplexer, die mehrere Datensignale mittlerer Bitrate (0,6 bis 9,6 kBit pro Sekunde) zu einem 48- oder 64-kBit/s-Signal zusammenfassen, dazwischengeschaltet.

Das ungarische Unternehmen Videoton war zum 20. Male in Leipzig vertreten. Es stellte erstmals eine Lösung zur Integration eines EC 1834 in ein **NOVELL-LAN** vor (siehe Bild 11). Als Hardware wird hierfür das ARC-NET (mit einer Übertragungsgeschwindigkeit von 2,5 MBit/s) oder das Ethernet (10 MBit/s) angeboten. Der 32-Bit-Mikrorechner VT 180 diente als Server. LAN-Teilnehmer können der VT 110, der VT 160 oder andere XT- bzw. AT-kompatible Rechner sein. Die Netzsoftware ist kompatibel zu Novell Advanced Netware 286 oder zu Novell System Fault Tolerant Netware 286. Sie enthält Funktionen wie kontrollierter Zugriff zu gemeinsamen Dateien, Anlegen von Sicherheitsduplikaten der Directories und Dateibeschreibungstabellen, parallele Behandlung von Magnetplatten sowie ständige Kontrolle und Fehlermeldung. Über einen Gateway-Dateitransfer kann eine Kopplung an das Ethernet-Netz des VT 32 und des R 11 erfolgen.

Gegenüber den LANs werden sich in Zukunft stärker die ISDNs durchsetzen (Integrated Services Digital Networks = Dienste integrierende digitale Netze). Sie bieten den Vorteil, daß sie in einem Netz (über eine Leitung) sowohl Datenkommunikation als auch Fernsprechen, Fernkopieren, Bildschirmtext und Teletext vereinen. Die Standard Elektrik Lorenz AG (SEL) Stuttgart bot das **SYSTEM 12 B**, eine Variante der öffentlichen digitalen Vermittlungsstelle SYSTEM 12 für das Büro, an (siehe Bild links unten). Bei 100 Nebenstellen beginnend, ist das SYSTEM 12 B bis weit über 3000 Nebenstellen in kleinen oder in größeren Schritten erweiterbar. Es erlaubt die Datenkommunikation hausintern zwischen PCs, zum öffentlichen ISDN oder zwischen Terminals und Hostrechnern. Die Datenkommunikation ist sehr leicht über die im SEL-Komforttelefon integrierten Schnittstellen V.24 und V.25bis in zwei Varianten möglich. Die Kopplung erfolgt erstens über eine V.24/V.25bis-Schnittstelle mit maximal 9600 Baud für PCs und asynchrone Terminals sowie über einen Koax-Adapter für 3270-Terminals. Zweitens ist die Nutzung der zwei 64-KBit/s-Kanäle als „Mehrdienste-Anschluß" möglich; sie erlaubt eine diensteunabhängige Mischkommunikation. Über Terminaladapter können die Teilnehmer mit V.24-, X.21- oder Koax-Schnittstellen angeschlossen werden.



In unserem Bericht von der Budapester Messe 1988 (s. MP 10/88) hatten wir gezeigt, wie ein A 7150 mit der Zusatzfunktion eines Fernschreibers ausgestattet werden kann. Die dazu von der ungarischen Firma Triton entwickelte Lösung, eine Anschlußsteckeinheit und das Softwarepaket Gepard-16, wurde zur LFM an den Ständen von Robotron und Triton gezeigt. Mit dem **Telex-PC** können durch Selbstwahl Telexteilnehmer im nationalen und internationalen Telexnetz erreicht werden. Daneben ermöglicht er ohne Beeinträchtigung dieser Dienstleistungen die Abarbeitung beispielsweise der üblichen Büroaufgaben.

Mit dem von Transcommerz angebotenen automatischen Telexprozessor **textMaster 1000** (Bild 12) der Firma MARK können bis zu zwei Terminals an eine Telexlinie angeschlossen werden. Als Terminal können dabei elektronische Schreibmaschinen, PCs oder auch lokale Datennetze Verwendung finden. Über das Zusatzgerät lassen sich Telexmitteilungen vom eigenen Terminal aus vorbereiten, absenden und empfangen, wodurch Zeit- und Kostenersparnisse erreicht werden. Der textMaster 1000 arbeitet dabei nach dem Speichervermittlungsverfahren (store and forward). Nachdem die Telexmitteilungen am Terminal geschrieben wurden, werden sie dem textMaster 1000 übertragen, der die Absendung automatisch steuert. Eingehende Telexe werden im textMaster 1000 bis zur Übertragung an die Terminals zwischengespeichert.

Der gleichen Aufgabenstellung dient das von Rank Xerox angebotene Telexmanagementsystem **TelexBox 3** (Bild 13). Auch die TelexBox dient als Telexschnittstelle für Schreibmaschine, Textverarbeitungsautomat oder PC und erlaubt das Senden und Empfangen im Hintergrund. Ebenso sind weitere benutzerfreundliche Funktionen wie Zwischenspeicherung, Kurzwahl, Rufwiederholung und anderes vorhanden.

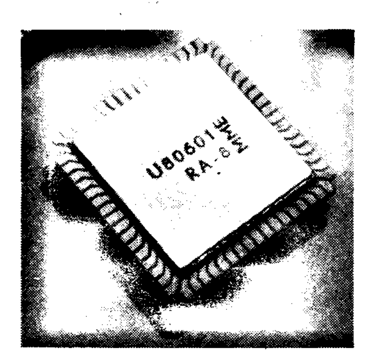
Bauelemente

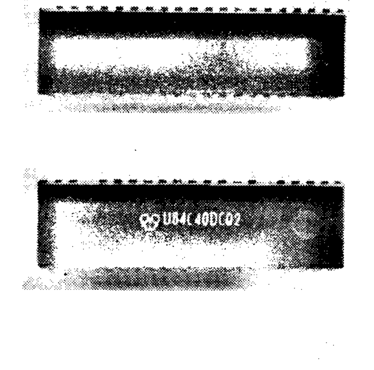
- **Mikroprozessorschaltkreise**

Das Mikroprozessorsystem U 80600 ist das zweite 16-Bit-System nach dem U 8000 aus dem Kombinat Mikroelektronik Erfurt. Es wird mit seinen konzipierten 20 Komponenten eine neue Generation von Personalcomputern und Steuerungssystemen in der DDR ermöglichen (zum Beispiel ICA 720). Die zur Frühjahrsmesse erstmals gezeigten vier neuen Typen dieses Systems sind die zentrale Verarbeitungseinheit (CPU), der Buscontroller, der DRAM-Controller sowie der Schaltkreis zur Fehlererkennung und -korrektur (EDC).

Die zum K 1810 WM86 (UdSSR) abwärtskompatible CPU **U 80601** (Bild unten) wird eine maximale Taktfrequenz von 16 MHz erreichen. Ihre Hauptverbesserungen sind die zwei Adressierungsmodi *real* und *protected* sowie der damit verbundene vollständige Speicherschutz als Voraussetzung für Multitasking. Die Statussignale der CPU werden vom Buscontroller **U 80606** dekodiert, der daraus die entsprechenden Lese- und Schreibkommandos für den Bus des Rechners erzeugt. Der DRAM-Controller **U 80610** vereinfacht den Aufbau von dynamischen Speichern, die aus Schaltkreisen mit 16, 64 oder 256 KBit bestehen können. Er verwaltet einen Adreßraum von maximal 2 MByte und eignet sich für die Ansteuerung von Dual-Port-RAMs. Zusammen mit dem EDC **U 80608** ermöglicht er das Erkennen und Korrigieren von Fehlern sowie einen einfachen Aufbau großer Speicherarrays. Diese vier Schaltkreise des Systems U 80600 haben wir bereits ausführlich in MP 5/1989 vorgestellt.

Neben der noch vorherrschenden NMOS-Technik setzt sich in der Mikroelektronik die CMOS-Technik mehr und mehr durch. Die klassischen Vorteile der NMOS-Technik – höherer Integrationsgrad, höhere Geschwindigkeit, geringerer Preis – werden mit der Weiterentwicklung der CMOS-Technik ausgeglichen. Die Mikroprozessorfamilie U 84C00 vom Kombinat Mikroelektronik Erfurt ist die erste 8-Bit-CMOS-Familie aus DDR-Produktion. In Leipzig zu sehen waren die Schaltkreise CPU **U 84C00** (Bild rechts oben), PIO **U 84C20**, CTC **U 84C30** und SIO **U 84C40** (Bild rechts Mitte) mit Taktraten von 2,5 MHz (zum Beispiel U 84C00 DC02) und 4 MHz (zum Beispiel U 84C00 DC04). Alle Schaltkreise dieser Familie sind software- und hardwarekompatibel zur Familie U 880. Hinzu kommen die Vorteile: geringerer Stromverbrauch und höhere Zuverlässigkeit. Die folgende Aufstellung zeigt einen Vergleich der Stromaufnahme (bei 4 MHz):



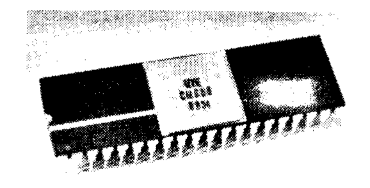


	U 84C00-Familie	U 880-Familie
CPU	25 mA	200 mA
PIO	5 mA	100 mA
CTC	7 mA	120 mA
SIO	15 mA	130 mA

Eine weitere Senkung des Stromverbrauchs wird durch den Schlafmodus erreicht, den die Schaltkreise des U 84C00-Systems einnehmen können, wenn keine Aktivität des Rechners nötig ist. Die Einnahme des Schlafzustandes wird von einem Taktgenerator/-controller (CGC) gesteuert.

Das System U 84C00 wollen wir in einem der nächsten Hefte ausführlich vorstellen.

Das Kombinat Mikroelektronik Botevgrad (VR Bulgarien) bot erstmals in Leipzig das 16-Bit-Mikroprozessorsystem SM 688 an. Es ist kompatibel zum Intel-System 8088. Unter anderem waren die Schaltkreise CPU **SM 688** (Bild unten), DRAM-Controller **SM 637**, Floppy-Controller **SM 609** und Harddisk-Controller **SM 610** ausgestellt.



Für die Verringerung des Platzbedarfs von Schaltkreisen setzt sich international der Einsatz von oberflächenmontierbaren Bauelementen (SMDs) durch. Dem trägt das Kombinat Mikroelektronik Erfurt mit der Vorstellung der 8-Bit-Einchiprechner **U 883** und **U 886** sowie des 64-KBit-DRAMs **U 2164** und des fensterlosen 32-KBit-EPROMs **U 2632** als SMDs Rechnung (Bild 14).

- **ASICs**

Gegenüber herkömmlichen Bauelementen in SMD-Ausführung läßt sich der Platzbedarf ganzer Schaltungen wesentlich effektiver durch den Einsatz von anwenderspezifischen integrierten Schaltkreisen (ASICs) – die wiederum SMDs sein können – verringern. Das Kombinat Carl Zeiss JENA bietet seit einigen Jahren zwei verschiedene ASIC-Typen an: das CMOS-Gate-Array-System U 5200 und den Standardzellenschaltkreis U 1500/20 (der U 1500 unterscheidet

sich vom U 1520 nur durch die Zahl der Verdrahtungsebenen). Während die Gate-Arrays auf vorgefertigten Siliziumscheiben eine feste Anzahl und Lage der Funktionselemente sowie feste Chipflächen und Pinzahlen verlangen, dafür aber schneller und billiger erhältlich sind, erlauben die Standardzellenschaltkreise eine variable Standardzellenzahl und -lage sowie unterschiedliche Chipflächen und Pinzahlen. Die nun vorgestellten Weiterentwicklungen der Systeme U 5200 und U 1500/20 sind das Gate-Array-System **U 5300** bzw. das Standardzellensystem **U 1600**. Die Entwicklung des Integrationsgrades dieser ASIC-Typen wird durch die Anzahl der Transistoren auf einem Chip verdeutlicht:

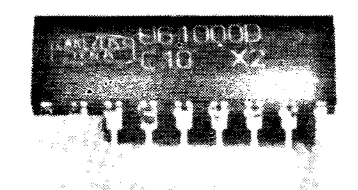
U 5200	12000	U 1500/20	13000
U 5301	40000	U 1600	100000
U 5302	70000		

Beide Systeme enthalten unter anderem Inverter (mit und ohne Treiberverhalten), Grundgatter (wie NAND, NOR, XOR, XAND und kombinatorische Verknüpfungen), Halb- und Volladder, JK- und D-Master-Slave-Flipflops sowie vielfältige Eingangs- und Ausgangsstufen mit und ohne D-Flipflop- oder Schmitttrigger- bzw. Tristate-Verhalten. Die Gatterverzögerungszeit beträgt bei beiden Systemen maximal 1,6 ns. Ein U 1600 erreicht damit eine Taktfrequenz von 25 MHz; seine Ruhestromaufnahme ist kleiner als 400 μA. Die Taktfrequenzen beim U 5300 sind abhängig vom Typ des Masters. Die Master U 5301 und U 5302 können mit 40 bzw. 30 MHz getaktet werden; ihre Ruhestromaufnahme liegt unter 200 μA. Als Entwurfssystem bietet Zeiss für die Gate-Arrays das System Archimedes an (siehe auch Teil 1 unseres LFM-Berichtes sowie MP 6/1989, Seite 168). Für die Standardzellen gibt es das Entwurfssystem ENSIC.

- **Speicherschaltkreise**

Bekanntlich unternimmt das Kombinat Carl Zeiss JENA große Anstrengungen bei der Entwicklung von dynamischen und statischen RAM-Schaltkreisen. Während auf der LFM '88 der 256-KBit-DRAM U 61256 in einer 1,5-μm-CMOS-Technologie Premiere hatte, konnte zur diesjährigen Frühjahrsmesse bereits der 1-MBit-DRAM **U 61000** (siehe Bild rechts oben) in einer 1-μm-CMOS-Technologie vorgestellt werden. Beide Schaltkreise werden zur Standardausrüstung moderner Personalcomputer und Workstations gehören. Der U 61000 wurde mit einem 18poligen DIL-Duroplastgehäuse gezeigt, kann aber auch in einem Keramikgehäuse geliefert werden. Er wird mit Zugriffszeiten von 100 und 120 ns verfügbar sein (U 61256: 80, 100, 120 und 150 ns). Seine Ein- und Ausgänge sind TTL- und CMOS-kompatibel. Er besitzt zur Zeit eine Organisation von 1 M × 1 Bit, und er kann in elf verschiedenen Betriebs- und Refresharten arbeiten. Der U 61000 ist zu internationalen Vergleichstypen wie TC 511000 (Toshiba) und HYB 511000 (Siemens) kompatibel. Im Heft 10/89 werden wir Ihnen diesen Schaltkreis näher vorstellen.

Damit große Speicherarrays möglichst platzsparend aufgebaut werden können, beschritt das Kombinat Keramische Werke Hermsdorf den Weg, sowohl 256-KBit-Chips als auch 1-MBit-Chips auf Keramiksubstrate zu montieren, um so Schaltkreise mit 4 MBit Speicherkapazität verfügbar zu machen. Der **4737** (Bild 15) enthält 16 Chips des Schaltkreises U 61256, die einen Speicher von 256 K × 16 Bit bilden, sowie je einen Keramik-Stützkondensator. Er besitzt ein 34poliges DIL-Gehäuse (Reihenabstand: 37,5 mm; Höhe: 5 mm; Pinraster: 2,5 mm) und wird nur mit einer Zugriffszeit von 150 ns angeboten. Eine andere Ausführung dieses Schaltkreises ist der **4743** mit einer Speicherorganisation von 512 K × 8 Bit. Dagegen ist der Schaltkreis **4742** in einer 25poligen SIL-Bauform (Bild 16) aus vier Chips des 1-MBit-RAMs U 61000 aufgebaut (mit ebenfalls je einem Kondensator). Damit ergibt sich eine Speichergröße von 1 M × 4 Bit.



Speicherschaltkreise waren auch am Stand des Kombinates Mikroelektronik Botevgrad ausgestellt: der 64-KBit-DRAM **SM 8164** (Organisation: 64 K × 1 Bit; Technologie: NMOS; Zugriffszeiten: 150 und 200 ns), der 4-KBit-SRAM **SM 8514** (1 K × 4 Bit; CMOS; 200, 300 und 450 ns) sowie der 64-KBit-EPROM **SM 7764** (8 K × 8 Bit; NMOS; 450 ns).

Software

Für die 8- und 16-Bit-Technik des VEB Kombinat Robotron wurde vom Robotron-Vertrieb Berlin das **P**ortable **OE**konomische Software-**SY**stem **POESY** vorgestellt. Es besteht aus den Komplexen Kostenrechnung, Lohn- und Gehaltsrechnung, Grundmittelrechnung, Investitionsrechnung, Materialrechnung, Leistungsrechnung, Nutzensrechnung und Rechnungsausgang. Alle Elemente dieses Systems können sowohl einzeln als auch kombiniert mit einigen oder allen anderen Bestandteilen eingesetzt werden. Dabei soll eine einfache Handhabung durch eine übersichtliche Programmstruktur, die menügesteuerte Auswahl der Programmteile, Fehlerermittlungen usw. gewährleistet sein. POESY soll ständig weiterentwickelt und vervollkommnet sowie an neue Technik angepaßt werden. Für den 8-Bit-PC 1715 unter dem Betriebssystem SCP, den A 7100 unter SCP 1700 und die 16-Bit-PCs A 7150 und EC 1834 unter DCP wird POESY angeboten; eine Beschleunigung der Arbeit durch eine RAM-Disk bei fehlender Festplatte wird unterstützt.

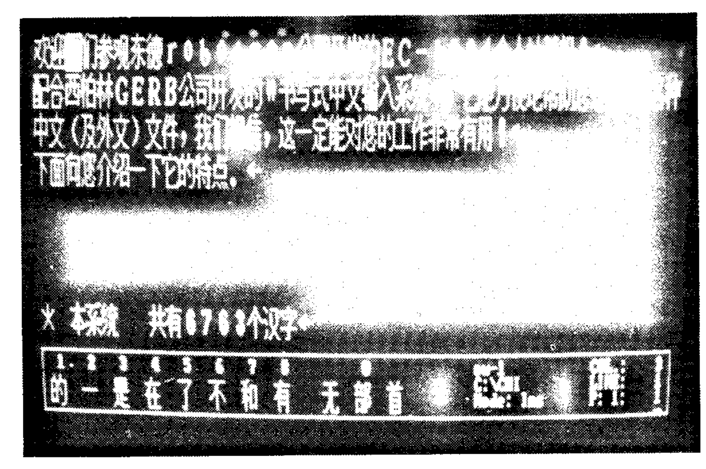
Der VEB Kombinat Robotron zeigte weiterhin die Einsatzmöglichkeiten des EC 1834 zur Textverarbeitung in arabischer und chinesischer Sprache. Das System **Al-BAYAN** gestattet unter dem Betriebssystem DCP das Editieren, Manipulieren, Gestalten und Ausdrucken von lateinischen und arabischen Schriftzeichen innerhalb eines Textes. Im *arabischen* Editiermodus werden die einzelnen Zeichen wie üblich von rechts nach links auf den Bildschirm geschrieben. Zur Menüführung kann wahlweise zwischen englischer und arabischer Sprache ausgewählt werden. Zahlreiche Funktionen für eine umfassende Textmanipulation und Textgestaltung gehören ebenfalls zum System AL-BAYAN.

Die chinesische Schrifteingabe wird auf PCs mit dem Betriebssystem MS-DOS durch das System **Chi-Easy** möglich (Bild unten rechts). Zur Eingabe der gewünschten Zeichen wird eine besonders codierte Zeichenvorlage benutzt, über die ein Lesestift wie beim normalen Schreiben des Zeichens geführt wird. Für die im Zeichensatz enthaltenen 6763 Zeichen sind jedoch maximal 5 bis 6 Striche erforderlich, da nach jedem neuen Strich 8 der am häufigsten vorkommenden Zeichen, die diese Striche enthalten, angezeigt und dann ausgewählt und auf den Bildschirm geschrieben werden. Nach dem zweiten Strich ist in 85 Prozent der Fälle das Zeichen gefunden. Die Druckausgabe der Texte ist ebenfalls möglich.

Die wissensgestützte Entwicklung problemspezifischer Software mit dem System **X...** ermöglicht die Erarbeitung von Programmen in verschiedenen Zielprogrammiersprachen unter Einbeziehung bereits vorhandener Programmbibliotheken. Durch die Modifikation oder den Austausch der eingesetzten Wissensbasis kann das System sehr leicht an unterschiedlichste Aufgabenstellungen angepaßt werden. Dem Softwareentwickler wird ein modernes und effektives Mittel zur computergestützten Softwareentwicklung in die Hand gegeben; es gestattet eine höchst flexible Nutzung der Anwendungssoftware durch den Endnutzer und stellt ein bequemes und leistungsfähiges Werkzeug zur Dokumentation, Wartung und Pflege der Software dar. Mit dem System X... kann sowohl der Nutzer im Programmentwicklungsmodus als auch der Programmierer im Editiermodus arbeiten. Zahlreiche Servicefunktionen, wie die grafische Darstellung der Programmstrukturen und die Einblendung verbaler und grafischer Hilfs- und Übersichtsfunktionen in Fenstertechnik, erleichtern die Arbeit. Das System X... wurde von der Akademie der Wissenschaften der DDR entwickelt und bisher erfolgreich in den Paketen Xamba (Erzeugung von Programmen für die Mikroskopbildauswertung), Xfortran (Erzeugung von Programmen für die Auswertung von Abbildungen des menschlichen Herzens) und Xdbase als einem intelligenten Datenbank-Interface eingesetzt.

Die Firma ABC-Computer Technik Pöstges, erstmals Aussteller in Leipzig, stellte ihr **ABC-System**, ein Kalkulationssystem für Druckereien vor. Mit diesem System ist es beispielsweise möglich, Kalkulationen für den Druck, den Satz und für Reproduktionen in kürzester Zeit zu ermitteln, die Verwaltung der Stammdaten und statistische Berechnungen vorzunehmen sowie Schriftverkehr (Druck von Angeboten, Auftragsbestätigungen, Lieferscheinen, Rechnungen, Mahnungen usw.) abzuwickeln. Als Basis für die Kalkulation eines Auftrags dient eine arbeitsplatzorientierte Stammdatei, aus der das System nach der Eingabe der technischen Personalkosten und der Gemeinkosten den Stundensatz je Arbeitsplatz berechnet. Weiterhin werden die Kosten für das Fertigungsmaterial eingegeben und daraus der Zeitaufwand für die jeweils erste und jede weitere Einheit ermittelt. Zu den Arbeitsplätzen können zusätzliche Parameter, beispielsweise für die Berechnung von Archivmontagen, angegeben werden. Die Vorkalkulation für den Druck, die kaum länger als 1 ... 3 Minuten dauert, kann ohne große polygrafische Kenntnisse ausgeführt werden, da das System ausgehend von dem vorhandenen Maschinenpark nach der Angabe einer Maschine weitere vorschlägt, drei Varianten mit unterschiedlichem Nutzen gleichzeitig kalkuliert und dabei Werte für Rohbogen, Montagen, Platten, Drucke und anderes automatisch berücksichtigt und den Nutzen grafisch darstellt. Aus dieser Vorkalkulation kann automatisch ein Angebot erstellt werden, das auf dem Bildschirm noch geändert werden kann. Die ebenfalls integrierte Artikelverwaltung ermöglicht eine exakte Übersicht über die Bestände und Bewegungen im Lager. Mit der Rechnungslegung werden die Bestände aktualisiert. Ebenso ist ein ständiger Überblick über die vorhandenen Papiersorten und über betriebswirtschaftliche Abrechnungen gegeben.

Wie bei vorangegangenen Messen in Leipzig bereits praktiziert, so wurden auch während der Frühjahrsmesse 1989 in der Karl-Marx-Universität Softwarelösungen vorgestellt, die im Rahmen wissenschaftlicher und studentischer Forschungsarbeit entwickelt wurden.



Fotos: Becker, Hemke, Weiß, Werkfoto

Die Sektion Technische Elektronik der Wilhelm-Pieck-Universität Rostock war mit dem Programmiersystem **comForth** vertreten und bietet damit eine portable und interaktive Umgebung zur Entwicklung prozeßorientierter Software in Forth an.
Das System comForth in der Version 2.xx enthält neben dem Basissystem nachladbare Komponenten, wie eine Programmsammlung zu Datentypen, Zahleneingaben usw., headerlose Hilfsdefinitionen für eine modulorientierte Programmierung sowie Bildschirm- und Tastaturanpassungen. Ein Wordstar-kompatibler Editor, Assembler und Debugger werden ebenfalls angeboten, wie auch weitere, spezifische Funktionen des jeweiligen Betriebssystems. Das System comForth wird ab sofort für CP/M (Z 80 und 8086) und MS-DOS (8086) angeboten. Erweiterungen, wie ein Numerikpaket (Fließkomma- und Matrixoperationen), das Multitasksystem für Aufgaben der Prozeßautomatisierung und die Cross-Compiler, sind in der Entwicklung und für einige Prozessoren bereits vorhanden.

Von der Karl-Marx-Universität Leipzig wurde das menügesteuerte Statistikpaket **PSYSTAT** für 16-Bit-PCs unter MS-DOS gezeigt; eine etwas abgerüstete Version des Pakets soll ab September 1989 auch für den A 7100 verfügbar sein. Ohne besondere Vorkenntnisse der Rechentechnik soll eine sofortige Anwendung durch die dialogorientierte Arbeitsweise ermöglicht werden. Umfangreiche Help-Unterstützungen, gezielt mögliche Nutzereingriffe, ständige Informationen über das aktuelle statistische Verfahren und die momentane Operation des Rechners zeichnen das Programm ebenso aus, wie die trotz der Vielfalt der statistischen Verfahren gegebene Erweiterungsmöglichkeit durch nutzereigene Algorithmen und eine grafische Datenpräsentation. Von den möglichen statistischen Verfahren seien hier nur einige erwähnt, wie beispielsweise Varianz, Varianzbreite, Rangplätze, Häufigkeiten, Prozentränge, Histogramme, multiple lineare Regression mit bis zu 50 Einflußgrößen und grafischer Ausgabe von Funktion und Punktwolke sowie mehrere Verfahren zur Faktoranalyse, zur Clusteranalyse und zur Diskriminanzanalyse.

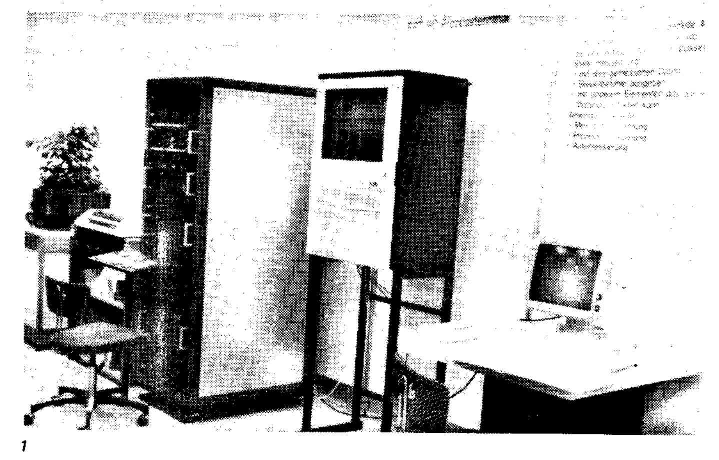
Das von dem Uppsala University Data Center (UDAC) in Schweden entwickelte und in der DDR von der Medizinischen Akademie „Carl Gustav Carus" vertriebene Datenbankbetriebssystem **MIMER** nimmt seine Kraft nicht nur aus dem Namen eines sagenhaften Weisen, sondern unter anderem aus der Anwendung der Standardabfragesprache SQL, die jetzt auch in dBase IV von Ashton-Tate integriert wurde. Besonders zum Verwalten großer Datenmengen ist MIMER durch schnelles Auffinden von Daten (Verwendung von Sekundärindizes) und das Speichern in einer ausgeglichenen Baumstruktur geeignet. Eine dynamische Entwicklung der Datenbanken durch Änderungen und Erweiterungen erfordert keine Änderung vorhandener Anwendungsprogramme, die zudem in den verschiedensten Sprachen (Fortran, Pascal, C, Lisp usw.) geschrieben sein können.

H. Hemke, H.-J. Hill, H. Weiß

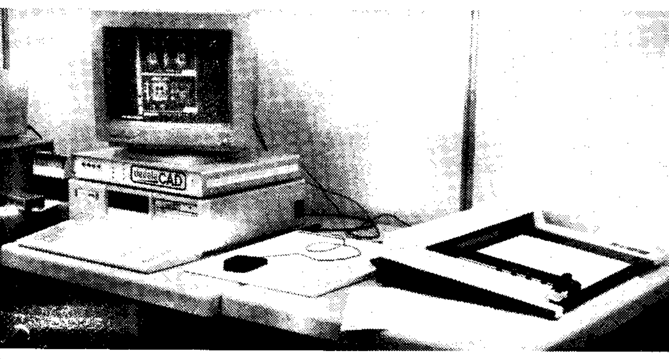
Neues von Videoton

Anhand zweier Veranstaltungen machten die ungarischen Videoton-Werke ihre Anwender und potentielle Kunden im Frühjahr dieses Jahres mit dem gegenwärtigen und künftigen Angebot bekannt. Da war zunächst das Videoton-Symposium Ende März in Klingemühle, das Interessierten in bewährter Weise Gelegenheit bot, in Vorträgen, aber vor allem auch in persönlichen Gesprächen mit Verantwortlichen, Einzelheiten zu erfahren. Neben Vorträgen zu Vernetzungsmöglichkeiten und grafischen Anwendungen der Videoton-Computer gab es interessante Ausführungen zur künftigen Entwicklung, die übrigens auch RISC-Systeme beinhalten wird. Ein Schwerpunkt waren die Automatisierungs- und die Steuerungstechnik. Informiert wurde unter anderem über die steuerungstechnischen Systeme MPT (Mini Process Terminal), RPT 90 (Remote Process Terminal), VT 32 C als Steuerungsrechner sowie die Variante VT 32 D als Programmentwicklungssystem für Steuerungsaufgaben. Das verstärkte Engagement Videotons auf diesen Gebieten zeigte sich ja bereits zur Leipziger Messe mit der vorgeführten Robotertechnik. Es wurde bekräftigt während einer Ausstellung, welche die Videoton-Werke Ende April in Berlin veranstalteten, auf der auch das RPT 90 zu sehen war (Bild 1). Das Prozeßterminal besteht aus einem 16/32-Bit-Mikroprozessor, Kommunikationsmodulen und der notwendigen Prozeßperipherie. Die Module entsprechen den Eurocardstandardkarten und können als Racks sowohl in 19-Zoll-Standardgestelle als auch in spezielle Geräteschränke mit Schutzklasse IP 41 montiert werden. Die Kommunikation erfolgt über einen seriellen Bus, der dem Intel-Bitbus entspricht.
Natürlich zeigte die Ausstellung auch Videotons Computerangebot: vom Minirechner R 11 über den VT 320 B, die Vernetzungsmöglichkeiten im Ethernet (anhand VT 32, VT 160 und R 11Y) und im Arcnet (anhand VT 110, VT 160, EC 1834 und VT 180 als Server) unter Novell NetWare bis zu Anwendungen für Konstruktion und Büroautomatisierung.
So etwa das zweidimensionale CAD-System dedataCAD für zahlreiche Einsatzgebiete (Bild 2), dessen extrem hohe Arbeitsgeschwindigkeit sowie der hohe Bedienkomfort besonders hervorgehoben wurden. Auf Lizenzbasis wird von Videoton das Programmpaket Intuitive Solution zur Vereinfachung der Büroarbeit angeboten. Gemeint ist die Verarbeitung von Texten und das Gestalten von Formularen mit einer leicht erlernbaren Bedienoberfläche anhand von Ikonen. Der obere Ausdruck zeigt beispielsweise den Ikonenbestand des Programms, der beliebig erweiterbar ist und selbstverständlich auch deutschsprachige Namen zuläßt. Unten ist das Beispiel für einen Kalkulator in Tabellenform zu sehen.
Mit dem Anschluß von Scanner und Laserdrucker (Bild 3) werden darüber hinaus bereits Merkmale des Desktop Publishing realisiert. *MP-We*

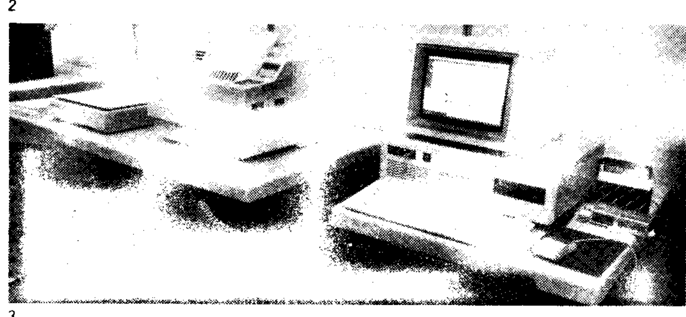
Fotos: Weiß



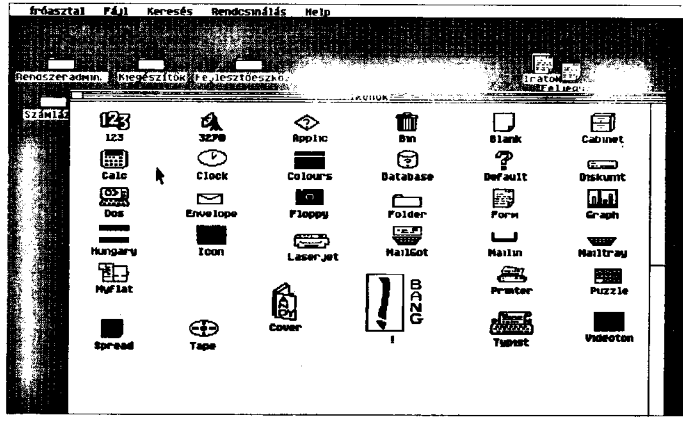
1

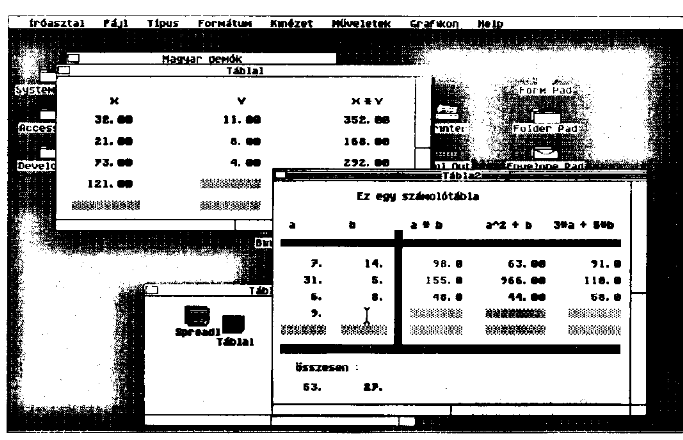


2



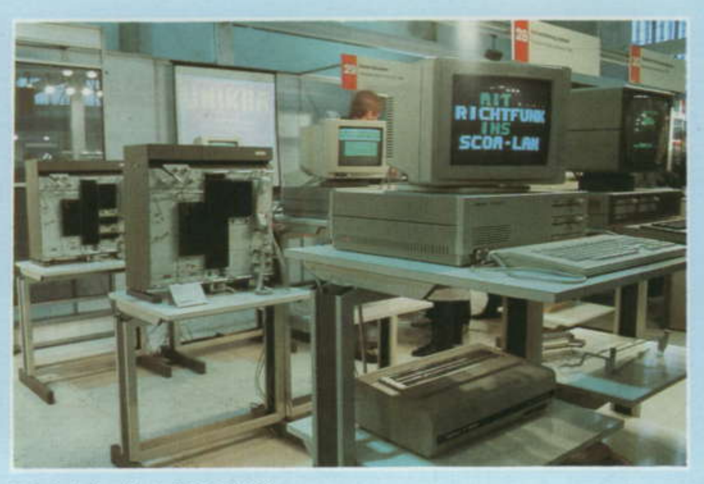
3







9 *DTP-Arbeitsplatz von Transcommerz*



10 *Lokales Netz SCOM-LAN*



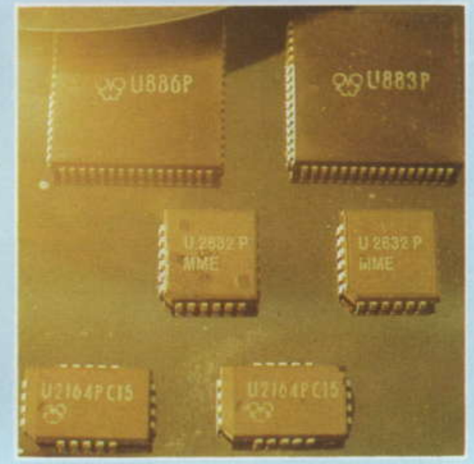
11 *Videoton-ARCNET*



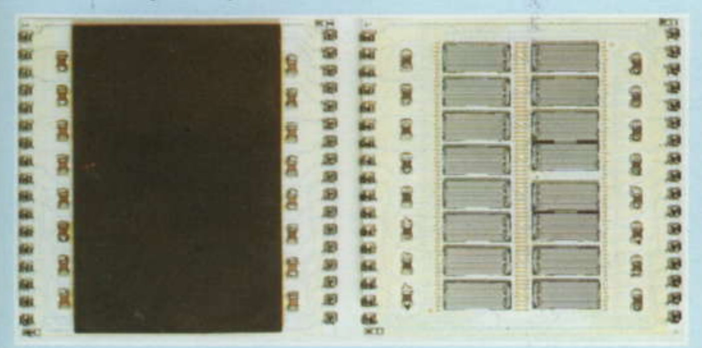
12 *Telexprozessor textMaster 1000*



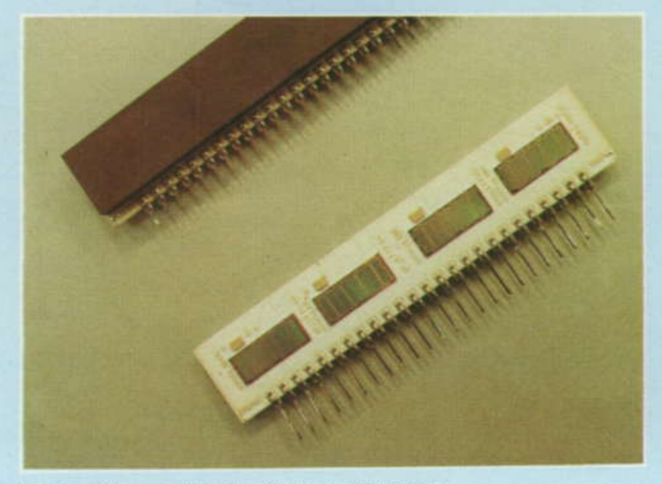
13 *Telexmanagementsystem TelexBox 3*



14 *Oberflächenmontierbare Schaltkreise*



15 *Speicherschaltkreis mit sechzehn 256-KBit-Chips*

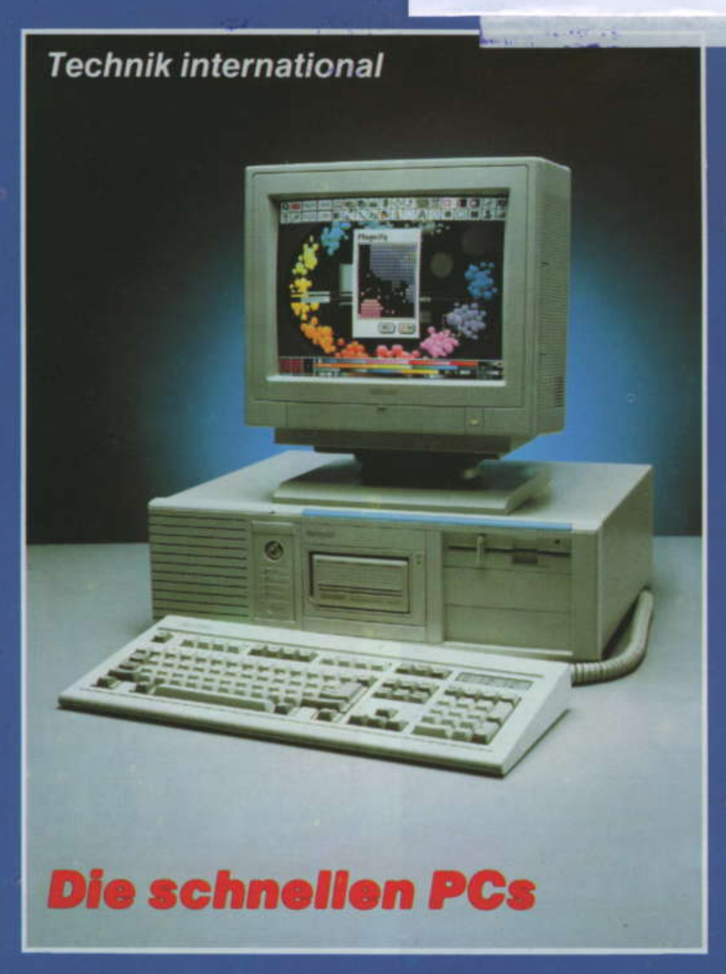


16 *Speicherschaltkreis mit vier 1-MBit-Chips*

ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 8
S. 225–256

Technik international

Die schnellen PCs



„Was du heute kannst besorgen, das verschiebe nicht auf morgen." Diese Spruchweisheit scheint in der Computerbranche seine besondere Berechtigung zu haben. Während wir uns im vergangenen Jahr immer wieder vornahmen, Ihnen, liebe Leser, einen Überblick über die damals neuen „Renner" bei den Personalcomputern (PCs) – mit 25-MHz-Prozessoren – zu liefern, hat uns die Entwicklung inzwischen überholt. Den Reigen der in der westlichen Welt so werbewirksamen 25-MHz-PCs eröffneten Mitte 1988 bekanntlich die Firmen IBM mit dem neuen Modell 70 des PS/2 und Compaq mit dem Deskpro 386/25 (siehe auch MP 9/1988; den 386/25 stellten wir in MP 2/1989 ausführlich vor). Zuvor hatte es allerdings schon Firmen gegeben, die stolz die „25" vorwiesen. Dabei handelte es sich aber um PCs, bei denen man sich die Qualitätsunterschiede in der Chipproduktion zunutze machte. Für 20-MHz-Prozessoren gibt der Hersteller zwar nur die Garantie einer fehlerfreien Funktion bis 20 MHz, aber aufgrund der Streuung in der Produktion können einzelne Exemplare durchaus auch bei höheren Taktfrequenzen fehlerfrei arbeiten. Dem ausgesuchten Prozessor wird nunmehr ein 25-MHz-Taktgeber verpaßt, und schon läßt sich mit der Spitzenfrequenz glänzen.
Auf dem gleichen – allerdings etwas fragwürdigen – Prinzip dürften auch die ersten Meldungen um die Jahreswende über 33-MHz-PCs beruhen, da Intel seinen 33-MHz-80386 erst seit kurzem ausliefert (dieser Beitrag hat selbstverständlich nur die PCs zum Inhalt, die nicht auf Motorola- oder Spezialprozessoren, sondern auf dem 32-Bit-Prozessor Intel 80386 basieren). Zu den Firmen, die nach eigenen Angaben „aufgebohrte" 25-MHz-Prozessoren in ihre 33-MHz-Maschinen einsetzen bzw. einsetzten, zählen beispielsweise CAS mit dem Modell CAS 3C33 und der Hersteller Everex. Dieser Wettlauf um Megahertz und MIPS zeigt, wie hart die PC-Hersteller heute um Spitzenpositionen kämpfen müssen. Dabei sollte auch einmal bedacht werden, daß Geschwindigkeit des Prozessors durchaus nicht mit Leistung des PCs

Megahertz nicht mit Leistung gleichsetzen

gleichgesetzt werden kann. Verfügt der Computer beispielsweise über die normale PC/AT-Architektur und langsame Speicherchips (DRAMs), muß der Prozessor bei Speicherzugriffen oft Leertakte einfügen (die bekannten Wartetakte oder wait states). Von großer Bedeutung ist natürlich auch die Zugriffsgeschwindigkeit der eingesetzten Festplattenlaufwerke. So kann ein PC mit 20-MHz-Prozessor, einer ausgeklügelten Mehrbus-Architektur, großem und schnellem Cache-Speicher und schnellen RAMs also durchaus eine höhere Gesamtleistung besitzen als ein 25-MHz-PC. Nichtsdestoweniger verfügen PCs mit schnellen Mikroprozessoren – und die genannten Nebenbedingungen berücksichtigt – über die besseren Voraussetzungen zur Erfüllung anspruchsvoller Aufgaben.
Nun also zu den neuen „Superschnellen", den 33-MHz-PCs. Als erste Firma, die nach eigenen Angaben den „echten" 33er von Intel verwendet, präsentierte Tandon zur CeBIT '89 das im Bild gezeigte Modell 386/33 unter dem Slogan, damit bis weit in den Bereich der Superminicomputer vorzudringen und dem professionellen Anwender fast schon „Mainframe-Power" auf dem Schreibtisch zu bieten. Der Prozessor läßt sich jedoch mittels Knopfdruck auch auf 8 MHz heruntertakten, um bei Bedarf volle Kompatibilität zur gesamten Standardsoftware der MS-DOS-Welt zu gewährleisten. Auch der Tandon 386/33 verfügt über leistungssteigernde Merkmale, wie sie heute bei Spitzen-PCs zum guten Ton gehören; beispielsweise den intelligenten Befehlsakzelerator, der den ROM-Inhalt beim Systemstart automatisch in den schnelleren Shadow-RAM lädt. Oder einen Cachespeicher, der standardmäßig 64 KByte SRAMs enthält. Die Leistungssteigerung durch Cachespeicher beruht bekanntlich darauf, daß sie als Zusatzspeicher zwischen dem aus dynamischen RAMs bestehenden Hauptspeicher und dem Mikroprozessor nach einem bestimmten Algorithmus „auf Verdacht" mit vermutlich als nächstes benötigten Daten oder Befehlen gefüllt werden und daß sie diese, da aus schnelleren statischen RAM-Bausteinen aufgebaut, der CPU auch schneller zur Abarbeitung bereitstellen können. Nur wenn die Information doch nicht im Cache enthalten ist, muß die CPU auf den Hauptspeicher zugreifen. Die „Trefferrate" liegt heute aber dank ausgeklügelter Algorithmen zumeist bei über 90 Prozent.

Tandon als erster im Ziel

Zur weiteren Leistungssteigerung sind auf der Hauptplatine des 386/33 Sockel für einen 80387- und einen Weitek-Koprozessor enthalten. Beide Koprozessoren können auch simultan arbeiten.
In der Grundausstattung hat der 386/33 einen RAM von 1 MByte, der sich mit 1-MByte-SIMM-Bausteinen auf der Grundplatine bis auf 24 MByte aufrüsten läßt; das heißt, es wird keiner der Erweiterungssteckplätze benötigt (1×8 Bit X-kompatibel, 4×16 Bit AT-kompatibel, 1×32 Bit; der Plattencontroller belegt davon einen 16-Bit-Steckplatz, Bildschirmadapter und Schnittstellenkarte je einen 8- oder 16-Bit-Steckplatz).
Den neuen Tandon gibt es in verschiedenen Ausführungen: Modell 386/33 als reines Diskettensystem ohne Festplattenlaufwerke, Modelle 386/33-90, 386/33-330 und 386/33-660 mit Festplattenkapazitäten von 90, 330 oder 660 MByte (ESDI-Controller). Grundausstattung sind ein 5,25-Zoll-Floppylaufwerk (optional 3,5-Zoll-Floppy) und ein 14-Zoll-Monochrom-Grafikbildschirm mit 720×348 Pixel (optional verschiedene Farbmonitore). Neben MS-DOS 3.3, MS-GW-Basic und Data-Pac-Uti... wird die grafische Benutzeroberfläche MS-Windows/386 standardmäßig mitgeliefert, mit der Multitasking- und Netzwerkbetrieb als Server möglich wird. Die von Tandon entwickelten Data-Pacs, die „mobilen Festplatten" (wir berichteten in MP 8/1988 ausführlich darüber), sind selbstverständlich Bestandteil des neuen Systems.
Auch die Firma Acer zeigte auf der CeBIT '89 im März ihren ersten 33-MHz-PC, den Acer 1100/33 – allerdings nur inoffiziell.

Das Spitzenfeld vergrößert sich

Bereits zur kurz darauf stattfindenden Computermesse Comdex in den USA war die Schar der Anbieter von 33-MHz-PCs schon auf über ein Dutzend angewachsen. Die Firma Advanced Logic Research Inc. (ALR) beispielsweise zeigte ihr neues Modell mit 7,4 MIPS Leistung als Flexcache 33/386-Z für den „Einsteiger" in Desktop-Ausführung und als Tower. Typisch ist die von ALR entwickelte Flex-Cache-Dualbusarchitektur mit 32-Bit-Cachebus (bzw. 64-Bit-Cachebus) und 32 KByte SRAM (bzw. 128 KByte SRAM).
Als weitere Vertreter seien genannt: AST mit dem Premium 386/33, die schon erwähnte Firma Acer mit drei Versionen des Modells 1100/33, die Firma Everex Systems mit dem Step 386/33, Philips Electronics mit dem P 3370 und Zenith mit dem Z-386/33.
Als bemerkenswert sei festgestellt, daß sich zu den Erstanbietern inzwischen auch Firmen zählen können, die nicht zu den Großen gehören und im Computergeschäft noch relativ unbekannt sind. Beispielsweise die österreichische Firma Theuretzbacher & Co., die ihren Power AT-386/33 von der Tochterfirma Future technologie in den USA produzieren läßt.

Bei 33 wird noch nicht gepaßt

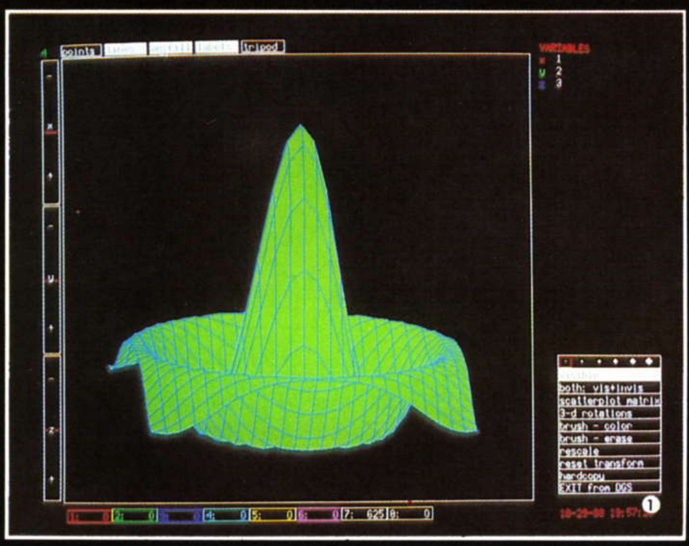
Der PC in Towerform ist AT-kompatibel, arbeitet mit 0 Wartetakten, besitzt einen von 2 auf 16 MByte erweiterbaren Hauptspeicher und einen beachtlichen 128-KByte-Cachespeicher. Als freie Steckkartenplätze gibt es 2×32-Bit-Slots. Neben 5,25- oder 3,5-Zoll-Floppies können Festplattenlaufwerke zwischen 40 und 676 MByte oder Streamer mit bis zu 150 MByte genutzt werden. Auch die übrigen Leistungsmerkmale, wie hochauflösende Grafikmonitore, Koprozessor usw., entsprechen durchaus dem in der Spitzenklasse gängigen Niveau.
Die Anstrengungen der Chiphersteller zur Geschwindigkeitssteigerung ihrer Prozessoren haben bei 33 MHz jedoch noch keineswegs ein Ende. Noch schnellere angekündigte Spezialprozessoren, aber auch Universalprozessoren, von Intels Konkurrenz entwickelt, dürfen vermuten lassen, daß auch beim 80386, aber zumindest beim Nachfolger 80486, noch höhere Taktraten zu erwarten sind. Womit wir in sicher nicht allzulanger Zeit etwas über 45- oder gar 50-MHz-PCs zu berichten haben werden.

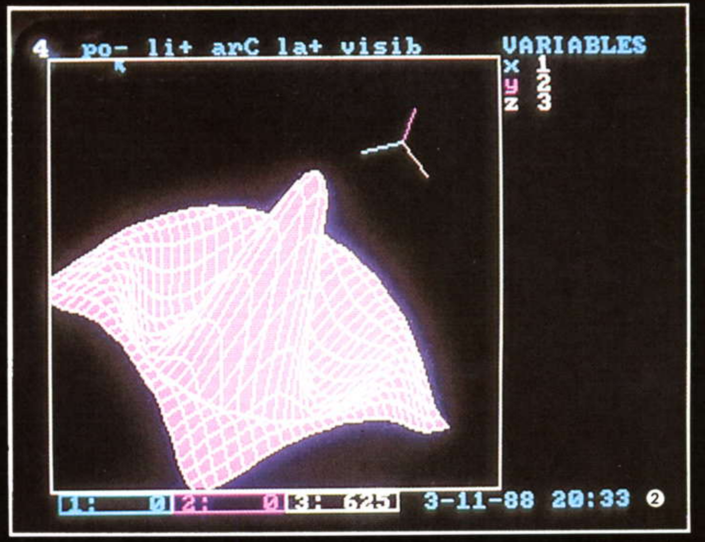
MP-We



Interaktive Echtzeitgrafik

Verwaltung von Zeichengeneratoren
Kursormanipulation
Heap-Nutzung unter Turbo-Pascal





Schnappschüsse von der rotierenden Funktion sin(r)/r. Während im EGA-/VGA-Modus 8 Farben gewählt werden können (Bild 1), stehen im CGA-Modus (Bild 2) lediglich 3 Farben zur Verfügung.

Interaktive Echtzeitgrafik

Lesen Sie auch unseren Beitrag auf der Seite 259.

Bei der Auswertung großer Datenmengen mehrerer variabler Einflußgrößen durch statistische Verfahren können grafische Darstellungen die Anschaulichkeit deutlich verbessern. Durch die Nutzung entsprechender Hard- und Software ist eine grafische Datenanalyse in einem dynamischen Umfeld möglich. Auf einem PC mit einem Hauptspeicher von mindestens 640 KByte, einem Farbbildschirm und einem Farbgrafikadapter (CGA, EGA oder VGA) ist es beispielsweise mit dem Programmsystem PC-ISP (Interactive Scientific Processor) auch unter MS-DOS möglich, grafische Darstellungen interaktiv in Echtzeit zu manipulieren.

So kann man beispielsweise dreidimensionale Objekte oder Punktwolken rotieren lassen. Die Bilder 1 und 2 zeigen die Funktion $z = \sin(r)/r$; dabei können mit der geringer auflösenden CGA-Karte nur Ergebnisse wie in Bild 2 erreicht werden.

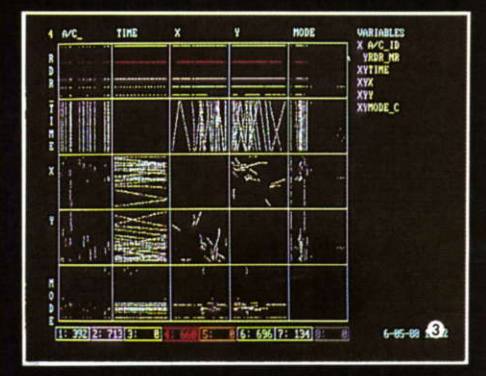
In der Streudiagramm-Matrix (Bild 3) werden die Flugkoordinaten (x, y), die Flughöhe (MODE-C), die Radar-Nummer (RDR.NR), die Flugidentifikation und die Zeit dargestellt. Die Daten stammen aus der zivilen Flugüberwachung, wobei die Flugzeuge von 5 Radarstationen überwacht werden. Ziel ist es, Störfälle am Radar aufzufinden und die Radarstationen unter anderem wie folgt zu bewerten:

- Welche Richtweiten haben die Stationen?
- Gibt es Reflexionen?
- In welchen Bereichen muß man den Radar abschalten, um Reflexionen auszuschließen?

Bild 4 zeigt die gleichen Daten als rotierende Punktwolke der Flugkoordinaten. Auffällig sind zwei Objekte, die von den Luftkorridoren (gepunktete Linie) abweichen. Diese lassen sich einfach identifizieren. Das Beispiel zeigt, wie umfassende Einzelinformationen mittels interaktiver Methoden und dynamischer Grafik schnell, bei Bedarf in Echtzeit, untersucht werden können.

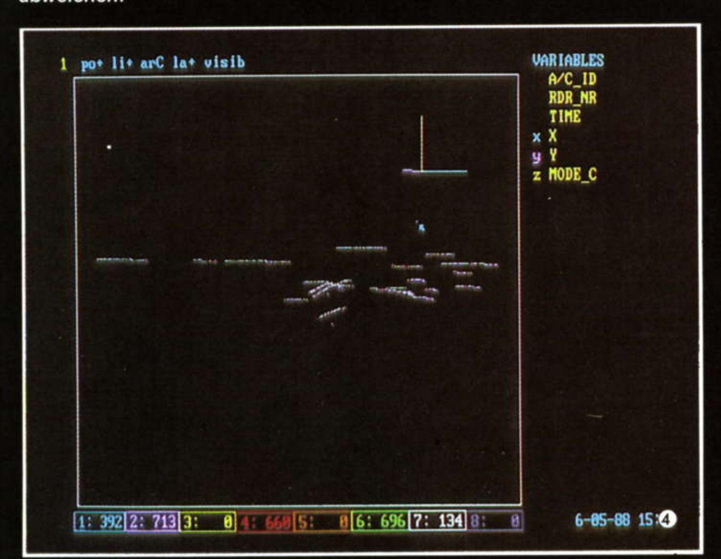
Im Bild 5 wird mit Hilfe der Matrixsprache ISP ein Zeitreihenmodell gesucht. Hier sind verschiedene Zeitreihenkurven überlagert. Bei PC-ISP/DGS steht dem Anwender ein Arbeitsbereich von 50 000 Real-Zahlen zur Verfügung, so daß damit eine unbehinderte und schnelle interaktive Arbeit mit etwa 15 000 Real-Zahlen möglich ist. Bei den 32-Bit-Rechnern der 80386-Serie erhöht sich der Arbeitsbereich auf 200 000 Real-Zahlen.

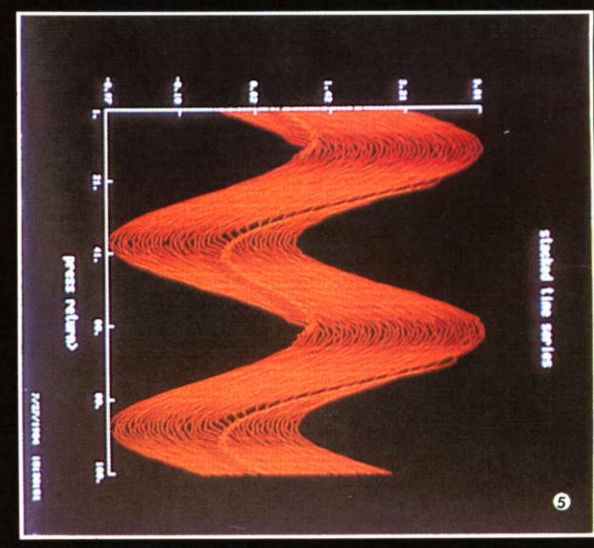
Fotos: Dr. M. Nagel und Th. Huber



Radardaten als Streudiagramm-Matrix. Dargestellt sind die Flugkoordinaten (x, y und Höhe in Fuß; 1 Fuß = 30,5 cm), die Zeit, die Flugnummer und die Radarnummer.

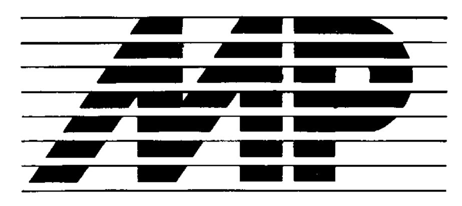
Radardaten. Für die Rotation wurden im Bild 4 die Flugkoordinaten (x, y und Höhe) ausgewählt. Es fallen dabei zwei Flugobjekte auf, die vom Luftkorridor abweichen.





Sehr einfach lassen sich auch Zeitreihen bearbeiten. Das Bild 5 zeigt mit Hilfe von PC-ISP überlagerte Zeitreihenkurven.

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 9 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Beirat Dr. Ludwig Claßen, Dr. Heinz Florin, Prof. Dr. sc. Rolf Giesecke, Joachim Hahne, Prof. Dr. sc. Dieter Hammer, Prof. Dr. sc. Thomas Horn, Prof. Dr. Albert Jugel, Prof. Dr. Bernd Junghans, Dr. Dietmar Keller, Prof. Dr. sc. Gernot Meyer, Prof. Dr. sc. Bernd-Georg Münzer, Prof. Dr. sc. Peter Neubert, Prof. Dr. sc. Rudolf Arthur Pose, Prof. Dr. sc. Dr. Michael Roth (Vorsitzender), Dr. Gerhard Schulze, Prof. Dr. sc. Manfred Seifart, Dr. Dieter Simon, Dr. Rolf Wätzig, Prof. Dr. sc. Dr. Jürgen Zaremba

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

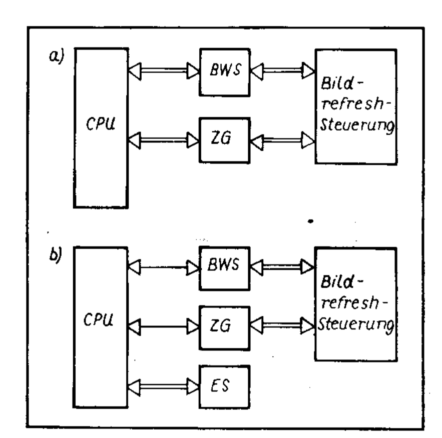
Redaktionsschluß 18. Juli 1989

AN (EDV) 49837

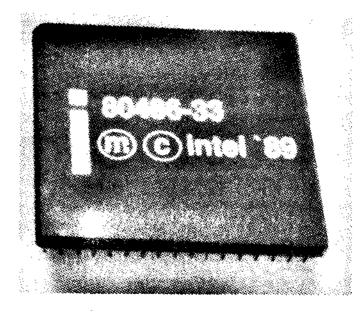
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tisku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Hà Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR–7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR–7010 Leipzig



Der Beitrag „Verwaltung programmierbarer Zeichengeneratoren" auf der **Seite 261** beschreibt pseudografische Verfahren für Rechner unter den Betriebssystemen SCP, UDOS und WEGA. Wegen des um etwa eine Zehnerpotenz niedrigeren Speicherbedarfs besitzen sie den Vorteil, daß fertige Bilder als topologische Strukturen abgespeichert werden können und damit eine minimale Generierungszeit bei Wiederaufruf gewährleistet wird.



Unter unserer Rubrik vorgestellt finden Sie auf der **Seite 278** den neuesten 32-Bit-Universalprozessor von Intel, den 80486. Er stellt eine Weiterentwicklung des 80386 dar und wird vorrangig in PCs und Workstations Verwendung finden.

Vorschau

In unserer Ausgabe zum 40. Jahrestag der DDR finden Sie Beiträge zu folgenden Themen:
- 1-Megabit-DRAM U 61000
- 8-Bit-CMOS-Prozessor U 84C00
- Kontamination
- Hybridschaltkreise

Inhalt

160 000 Personalcomputer: Parteitagsauftrag vorfristig erfüllt

Der 160 000. im Kombinat Robotron seit 1986 produzierte Personalcomputer wurde am 12. Juni an ein Jugendkollektiv des stadtgeleiteten Kombinates Bau und Modernisierung Dresden übergeben.
Mit der Übergabe des Jubiläumsrechners vom Typ A 7150 erfüllten die Kollektive des Kombinates Robotron das vom XI. Parteitag der SED 1986 gesetzte Ziel, 160 000 bis 170 000 Personal- und Bürocomputer bis 1990 zu produzieren. Die obere Grenze des Vorhabens werde bis zum 40. Jahrestag der Gründung der DDR erreicht, teilte der Stellvertreter des Generaldirektors des Kombinates Robotron, Dr. Egon Heusing, während der Übergabe mit. Die bis 1990 geplante Produktion von Personalcomputern kann damit wesentlich überboten werden. Der stellvertretende Generaldirektor dankte den Zulieferern unter anderem in den Kombinaten Mikroelektronik, Elektronische Bauelemente und Carl Zeiss JENA sowie allen anderen Kooperationspartnern. Mit den nahezu 68 000 Robotron-Werktätigen seien in der DDR rund 100 000 Werktätige an der Produktion von Computern beteiligt.
Dr. Heusing kündigte an, daß die von der Initiative der Sömmerdaer Büromaschinenwerker 1986 zur zusätzlichen Produktion von 10 000 PC 1715 über die Verpflichtung des Dresdner Stammbetriebes 1987 zur Verfünffachung der AC-Produktion reichende stetige Steigerung in der Computerfertigung mit der ständigen Neu- und Weiterentwicklung moderner Rechentechnik einhergehe. Dazu gehöre die Arbeit an neuen Generationen von Personalcomputern *ADN*

Neue Anlage für Megabit-Chips in Serie

Die Serienfertigung des automatischen Überdeckungsrepeaters AÜR 2 hat im Juni im Betrieb für optischen Präzisionsgerätebau des Kombinates Carl Zeiss JENA begonnen. Die Geräte werden in der mikroelektronischen Industrie für den Aufbau der Produktion monolithischer Speicherschaltkreise des Integrationsniveaus 1 Megabit benötigt. Mit dem AÜR 2 lassen sich außerdem Vorlaufarbeiten für die 4-Megabit-Technologie ausführen. Überdeckungsrepeater gelten als eine der bestimmenden Ausrüstungen im mikrolithographischen Fertigungsprozeß, der etwa 450 technologische Schritte auf rund 70 unterschiedlichen Geräten umfaßt. Der AÜR 2 überträgt die komplizierten Bauelementestrukturen der künftigen Schaltkreise optisch von Schablonen auf Siliziumscheiben. Dabei werden die Strukturen gleichzeitig um vier Fünftel verkleinert. Dieser Vorgang wiederholt sich im Automatikbetrieb pro Scheibe ungefähr zwanzigmal. Bei einem Bildfeld von 16 mal 16 Millimetern können Strukturen beispielsweise mit Leiterbahn-„Breiten" von nur einem tausendstel Millimeter realisiert werden. Die Überdeckungsgenauigkeit von Ebene zu Ebene wird von Fachleuten mit plus/minus einem zehntausendstel Millimeter angegeben. *ADN*

Treffen der Vorsitzenden der Paritätischen Regierungskommission DDR/UdSSR

Auf Einladung des Stellvertreters des Vorsitzenden des Ministerrats der DDR und Vorsitzenden des DDR-Teils der Paritätischen Regierungskommission, Gerhard Schürer, weilte Anfang April der Stellvertreter des Vorsitzenden des Ministerrats der UdSSR und Vorsitzende des sowjetischen Teils der Paritätischen Regierungskommission, Iwan Silajew, zu einem Besuch in Berlin.
Die Vorsitzenden erörterten Aufgaben in Vorbereitung der nächsten Tagung der Paritätischen Regierungskommission. In Verwirklichung der prinzipiellen Vereinbarungen, die im September 1988 getroffen wurden, standen insbesondere Maßnahmen der Zusammenarbeit in Wissenschaft und Technik für eine längerfristige Periode im Mittelpunkt. Gerhard Schürer und Iwan Silajew berieten dabei vor allem über Möglichkeiten, die Zusammenarbeit auf dem Gebiet der Schlüsseltechnologien auszubauen und darüber, wie die Koordinierung der Volkswirtschaftspläne für den Zeitraum 1991 bis 1995 weitergeführt wird. Gesprächsgegenstand war ebenso die Vertiefung der Produktionskooperation zwischen den Kombinaten und Betrieben der DDR sowie den Vereinigungen und Betrieben der UdSSR.
Die Maßnahmen sind darauf gerichtet, die Leistungskraft der Volkswirtschaften beider Länder entsprechend den Beschlüssen des XI. Parteitages der SED und des XXVII. Parteitages der KPdSU zielstrebig zu entwickeln und dabei die Potenzen der Spezialisierung und Kooperation immer umfassender zu nutzen. Große Aufmerksamkeit wurde progressiven Formen der Zusammenarbeit, wie den Direktbeziehungen auf der Ebene der Kombinate und Betriebe, gewidmet.
Während der Beratungen sind weitere Aufgaben der gegenseitigen Beziehungen behandelt worden, darunter zur Realisierung der im Februar 1989 unterzeichneten Konzeption der wirtschaftlichen und wissenschaftlich-technischen Zusammenarbeit zwischen der DDR und der UdSSR bis zum Jahre 2000 sowie des Komplexprogramms des wissenschaftlich-technischen Fortschritts der Mitgliedsländer des RGW.
An der Unterredung, die in einer freundschaftlichen Atmosphäre und der vollen Übereinstimmung der Ansichten verlief, nahmen seitens der DDR der Minister für Elektrotechnik und Elektronik, Felix Meier, und der Staatssekretär im Ministerium für Schwermaschinen- und Anlagenbau, Helmut Dersch, seitens der UdSSR der Minister für elektrotechnische Industrie, Oleg Anfimow, der Minister für Gerätebau, Automatisierungsmittel und Steuerungssysteme, Michail Skabardnja, und der Minister für Schwer-, Energie- und Transportmaschinenbau, Wladimir Welitschko, sowie weitere Persönlichkeiten teil. *ADN*

Erfolgreicher Messeabschluß für Robotron in Poznan

Die Internationale Messe für Elektronik, Fernmelde- und Computertechnik „Infosystem" in Poznan ging Mitte April erfolgreich zu Ende. Zu den 300 Ausstellern aus 15 Ländern gehörte auch das DDR-Kombinat Robotron, das die Exposition vor allem zur weiteren Markteinführung seines 32-Bit-Rechners K 1840 in Polen nutzte. Im Ergebnis der Leistungsschau wurde die Lieferung einer dieser leistungsfähigen Ingenieurarbeitsstationen für den Schaltkreisentwurf und die Leiterplattenproduktion sowie für den Einsatz als Steuerrechner zur Automatisierung der Produktion mit polnischen Partnern vereinbart. Das Kombinat liefert 1990 ebenfalls elektronische Schreibtechnik nach Polen. *ADN*

Gespräch Frankreich–DDR

Dreitägige Gespräche über die Möglichkeiten der Zusammenarbeit auf verschiedenen Gebieten der Spitzentechnologien sind Ende April in Paris zwischen Generaldirektoren von sieben Industriekombinaten der DDR und Präsidenten beziehungsweise Generaldirektoren führender französischer Unternehmen beendet worden. Die DDR-Delegation unter Leitung des Stellvertreters des Ministers für Außenhandel Christian Meyer weilte auf Einladung des Ministers für Industrie und Territorialgestaltung, Roger Fauroux, in Frankreich. Der Delegation gehörten die Generaldirektoren der Kombinate Nachrichtenelektronik Berlin, Jürgen Apitz, Automatisierungsanlagenbau Berlin, Heinz Brandt, Haushaltgeräte Karl-Marx-Stadt, Dr. Reiner Krannich, TAKRAF Leipzig, Kurt Schönfeld, „7. Oktober" Berlin, Dr. Heinz Warzecha, Mikroelektronik Erfurt, Prof. Dr. Heinz Wedler, und Robotron Dresden, Friedrich Wokurka, an.
Während einer Beratung mit der DDR-Delegation unterstrich Minister Fauroux das Interesse seiner Regierung am weiteren Ausbau der Wirtschaftsbeziehungen mit der DDR. Er verwies auf die große Bedeutung der Gespräche, insbesondere bei der Erschließung neuer Gebiete der industrielltechnischen und wirtschaftlichen Zusammenarbeit.
Der französische Minister betonte die Nützlichkeit der direkten Kontakte zwischen den Wirtschaftsexperten und Technikern beider Länder, da von gegenseitigen Kenntnissen über neue Technologien wichtige Impulse für einen verstärkten Warenaustausch ausgehen. Zu diesem Thema wurde ein Meinungsaustausch mit dem französischen Minister für Außenhandel, Jean-Marie Rausch, geführt.
Im weiteren Verlauf ihres Aufenthaltes besuchte die DDR-Delegation bedeutende französische Konzerne und Forschungseinrichtungen. In bilateralen Gesprächen wurden weitere Schritte zum Ausbau der ökonomischen Zusammenarbeit beraten.
Im Nationalen Forschungsinstitut für Informatik und Automatisierung (INRIA) wurden die Gäste aus der DDR mit neuesten Entwicklungsrichtungen und -programmen in der industriellen Automatisierung und Informationsverarbeitung bekannt gemacht. *ADN*

Österreich forciert Schlüsseltechnologien

Der verstaatlichte österreichische Elin-Konzern hat Mitte April im Wiener Industriebezirk Floridsdorf die erste Stufe eines Technologieparks in Betrieb genommen. Auf dem fast 180 000 Quadratmeter großen Gelände wurden nahezu alle Elektronik-Aktivitäten des Unternehmers mit den Schwerpunkten Automatisierungs-, Fernwirk- und Prozeßtechnik, Komponenten der elektrischen Antriebstechnik, der Leistungselektronik, der Kommunikationstechnik sowie die Entwicklung von Basistechnologien konzentriert.
Nach den Worten von Generaldirektor Guido Klestil stellte das Unternehmen damit die Weichen für die 90er Jahre. Die keineswegs neue und auf anderen Gebieten auch schon in Österreich praktizierte Grundidee bestehe darin, eigene Kapazitäten mit denen anderer Unternehmen zum gegenseitigen Vorteil zu verbinden, wozu auch gemeinsame Einrichtungen der sozialen Infrastruktur gehören. So haben sich in dem Technologie-Park bereits das Unternehmen Alcatel, mit dem Elin auf dem Gebiet der Künstlichen Intelligenz und der Software-Entwicklung zusammenarbeitet, sowie der USA-Computerhersteller Tandon mit einer Monatsmontage von 15 000 Personalcomputern für den europäischen Markt angesiedelt. Weitere Partner, die den Anforderungen des Elin-Konzepts entsprechen, sollen folgen und die Zahl der Beschäftigten von derzeit 1 400 auf 4 000 erhöhen. Bisher wurden innerhalb von drei Jahren 300 Millionen Schilling (rund 42,5 Millionen Mark) in Floridsdorf investiert, weitere 100 Millionen sind für die Errichtung einer Halle zur Fertigung von Personalcomputern vorgesehen.
Mit der Errichtung des Technologiekomplexes reiht sich Elin in die Bemühungen der österreichischen Wirtschaft ein, speziell im Bereich der Schlüsseltechnologien ein Forschungs- und Produktionspotential mit internationalem Niveau aufzubauen. Daß man sich mit dem erreichten Stand nicht zufriedengibt, dokumentierte unlängst der Wiener Professor Gerfried Zeichen. Er prophezeite Österreich eine „düstere Zukunft", wenn der Forschung nicht ein höherer Stellenwert eingeräumt und vor allem die Mikroelektronik nicht in breiterem Ausmaß in der Volkswirtschaft eingesetzt werde.

ADN-Preißler

Interaktive Datenanalyse und dynamische Grafik

Dr. Matthias Nagel
Forschungsinstitut für Hygiene und Mikrobiologie Bad Elster
Thomas Huber
Datavision AG, Klosters (Schweiz)

Empirische Forschungsarbeit läßt sich grob in zwei Phasen einteilen, eine experimentelle Phase, in der Erkenntnisse oder Hypothesen über den Gegenstand des Interesses aus vorliegenden Daten und über deren Zusammenhänge aufgestellt werden, und in eine daran anschließende bestätigende oder befestigende Phase, in der die hypothetischen Modelle überprüft werden müssen.

Sind die Datenstrukturen unbekannt, ist gerade das Aufstellen von Hypothesen und das Formulieren von Modellen *das* Problem, so daß es erforderlich ist, Daten und deren empirischen Hintergrund gründlich zu untersuchen – eine Aufgabe der explorativen Datenanalyse /1/, /2/. Grafische Methoden spielen hierbei zunehmend eine Rolle, da sie einerseits dem menschlichen Wahrnehmungssystem entsprechen:

- Durch die globale Betrachtung sind in Grafiken wesentliche Strukturen in den Daten schneller und leichter als in Tabellen und Texten zu erfassen.
- Details von Datenmengen können weit besser und leichter aufgedeckt, veranschaulicht und betont werden als in Tabellen.

Andererseits ist die Renaissance grafischer Methoden durch die Hard- und Software bedingt. Ebenso wie heute große Datenbestände direkt am Arbeitsplatz des Ingenieurs oder Wissenschaftlers verfügbar sind, lassen sich hier auch unaufwendig Grafiken zur Exploration, Analyse und Präsentation von Daten erstellen /3/. Waren bisher Grafiken gewissermaßen ein Endprodukt und zweidimensionale (2D) *Schnappschüsse* des p-dimensionalen Raumes, ist es nunmehr möglich, Grafiken in Echtzeit zu erzeugen *und* zu manipulieren.

Das bezieht sich nicht nur auf die Gestaltung, sondern ermöglicht auch ein Experimentieren mit den Daten: Der Analytiker führt mit einem Eingabegerät (Tastatur, Maus, Lichtstift) eine Operation aus – wählt beispielsweise eine Transformation der Daten aus – und (scheinbar) gleichzeitig wird diese auf dem Bildschirm in der Veränderung der Grafik sichtbar. Die **direkte Manipulation** grafischer Elemente auf dem Bildschirm und die (scheinbar) kontinuierliche Änderung dieser Elemente sind die wichtigsten Eigenschaften dynamischer oder interaktiver Grafik, die zukünftig die Datenanalyse entscheidend beeinflussen werden. Man sieht mehr, wenn man eine Grafik interaktiv beeinflussen kann, als wenn man sie lediglich betrachtet /4/. Dynamische Techniken vermitteln den Eindruck eines 2-D- oder 3-D-Films. Man kann sich das am besten mit der *glatten* Bewegung einer Punktwolke veranschaulichen: Es ist möglich, die Spur markierter Objekte zu verfolgen, Cluster und Ausreißer zu identifizieren und anderes mehr.

Dynamische Grafik

Die Ideen der Nutzung von Echtzeitgrafik für die Analyse von Daten sind nicht neu, allerdings handelte es sich bei den frühen dynamischen Systemen um aufwendige Prototypen, meist mit einem speziellen Hardware-Unikat. Als Vorbild für die heutigen Softwareentwicklungen zur dynamischen Grafik dient das zur Analyse von Teilchenströmen am Linearbeschleuniger in Stanford (USA) entwickelte System PRIM-9 /5/, wobei PRIM für einige wichtige, damit realisierbare Grundoperationen – Projektionen, Rotationen, Isolation bzw. Identifikation und Markierung bzw. Maskierung von Daten mit bis zu 9 Dimensionen – steht.

Durch Rotation einer Punktwolke wird ein räumlicher Eindruck der Datenstruktur vermittelt. Nach Zuordnung von Variablen zu den x-, y- und z-Koordinaten wird eine Rotationsachse ausgewählt und die 3-D-Punktwolke in die Bildschirmebene projiziert. Ist die z-Achse beispielsweise vertikal zum Bildschirm, dann entspricht die Rotation um die x-Achse einer Linearkombination der y- und z-Achse. Der neue 2D-Unterraum wird dann durch die alte x-Achse und die neue y-Achse definiert:

$$y(t) = y \cdot \cos(t) + z \cdot \sin(t), \quad 0 \leq t \leq \frac{\pi}{2}$$

Nun werden kontinuierlich für einen vorgegebenen kleinen Winkel t neue Projektionen auf dem Bildschirm abgebildet und durch die Bewegung der auf dem Bildschirm erscheinenden Punktwolke wird die Illusion einer dritten Dimension vermittelt. Rotationsgeschwindigkeit, Drehrichtung und Variablenauswahl lassen sich interaktiv durch den Anwender wählen, die Punktwolke kann vergrößert, verkleinert und auf dem Bildschirm verschoben werden. Bild 1 zeigt einen *Schnappschuß* einer solchen Rotation. Es handelt sich dabei um toxikologische Daten, bei denen an 48 Versuchstieren 17 Variablen beobachtet wurden.

Das Dreibein verdeutlicht die aktuelle Lage des Koordinatensystems mit den im Menü ausgewählten Variablen, die den Koordinaten zugeordnet sind. In dem Bild werden indirekt weitere dynamische Methoden verwendet. Ein wichtiges Hilfsmittel ist die Identifikation von Beobachtungen. Im Bild 1 fallen beispielsweise die 2 Beobachtungen in der Nähe des Dreibeins auf, die wie *Satelliten* die Punktwolke umkreisen. Möchte man diese ausreißerverdächtigen Beobachtungen identifizieren, kann man das tun, indem man den Cursor in ihre Nähe bewegt. Dadurch wird die Beobachtungsnummer oder der Name angezeigt (hier: Nr. 37 und Nr. 7). Umgekehrt ist es häufig von Interesse, bestimmte Beobach-

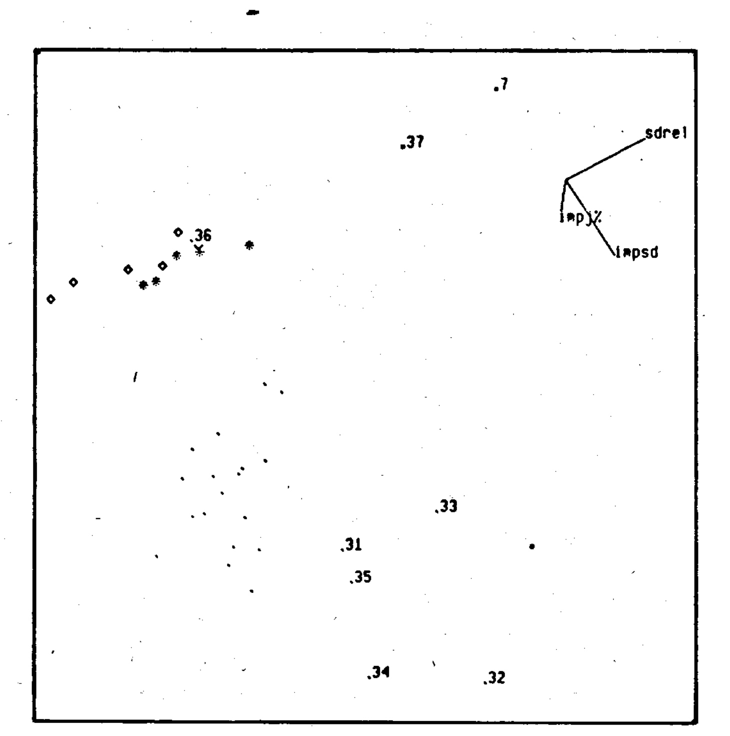


Bild 1 Schnappschuß einer rotierenden Punktwolke (Ausschnitt)

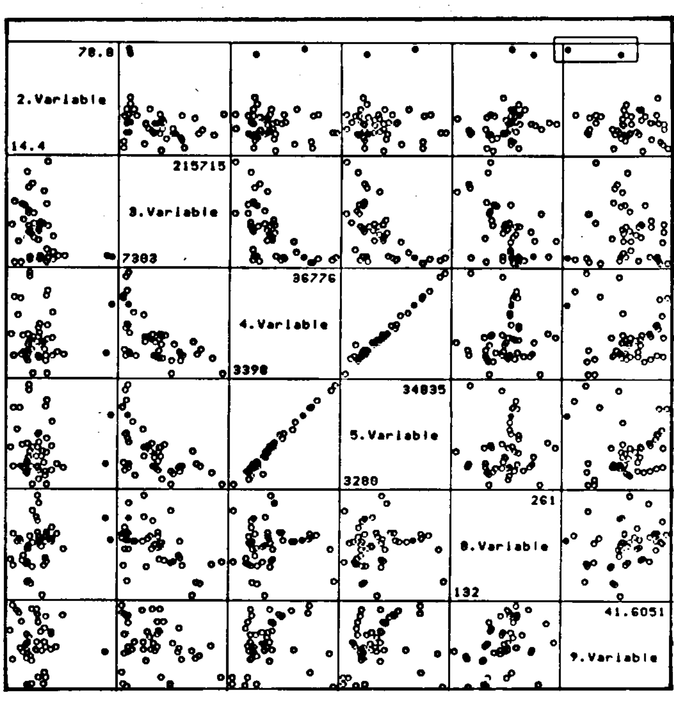


Bild 2 Streudiagramm-Matrix mit Brushing-Rahmen (Ausschnitt)

tungen im Datensatz wiederzufinden. Dazu kann man in einem Beobachtungsmenü die gewünschte Beobachtung auswählen, die dann sofort auf dem Bildschirm durch Blinken, Farbe oder ein Symbol hervorgehoben wird. Schließlich wurde im Bild 1 ein Objektcluster identifiziert und die Beobachtungen einer speziellen, vorgegebenen Gruppe (der Kontrollgruppe des Experimentes) durch ein spezielles Symbol (auch Farbe ist möglich) markiert. Auf diese Weise markierte und identifizierte Beobachtungen lassen sich in speziellen Teilmengen ablegen und bei Bedarf aus dem Bild ausblenden. Das Bild wird danach neu aufgebaut und umskaliert. Solche Teilmengen lassen sich in der Grafik auch durch *Anklicken* einzelner Beobachtungen mit dem Cursor oder durch Einschließen in Polygonzüge, im Beobachtungsmenü durch Auswahl von Beobachtungen, erzeugen, getrennt analysieren (und dabei jederzeit wieder in den Datenbereich aufnehmen).

Neben den Punkten lassen sich auch Linien und Flächen an das rotierende System übergeben. In den Farbbildern 1 und 2 auf der 2. Umschlagseite rotiert beispielsweise die Funktion

$$z = \sin(r) \text{ mit } r = \sqrt{\frac{x^2 + y^2}{1{,}5}}$$

und die einzelnen Gitterpunkte wurden durch Flächen gefüllt. Bei der Rotation werden aus rechentechnischen Gründen allerdings nur die Gitterpunkte als Skelett bewegt. Wird diese Rotation unterbrochen, wird das Bild mit Flächen bzw. Linienzügen wieder vollständig aufgebaut.

Bild 2 zeigt eine weitere interaktive Methode: In einer Streudiagramm-Matrix werden ausgewählte Variablen gegeneinander aufgetragen. So entspricht beispielsweise das Feld (1,2) dem Streudiagramm, das durch die Variablen 2 und 3 gebildet wird. Auch hier lassen sich Beobachtungen identifizieren. Fährt man mit dem Cursor ein Objekt in einem Streudiagramm an, wird es zeitgleich durch Blinken oder Einfärben in allen anderen Feldern der Streudiagramm-Matrix hervorgehoben. Wiederum lassen sich Gruppen markieren und farbig hervorheben wie es im Titelbild zu sehen ist. Von R. A. Fisher wurden hier die drei Iris-Arten (Schwertlilien) Setosa, Versicolor und Virginica analysiert und durch die Variablen Länge und Breite der Kelchblätter sowie Länge und Breite der Blütenblätter charakterisiert. Von Interesse ist, wie die Arten durch diese Variablen unterschieden werden können. So hebt sich beispielsweise die Länge der Blütenblätter (Variable petale Länge) der Art Iris setosa (rot) deutlich von der anderer Arten ab.

Eine mögliche visuelle Verknüpfung zwischen den einzelnen Variablen kann durch das Brushing erreicht werden. Brush (deutsch: Bürste) ist ein in Größe und Form frei bestimmbarer rechteckiger Rahmen, der mit Hilfe der Cursortasten oder der Maus über den Bildschirm bewegt werden kann. Punkte, die in dem Rahmen liegen, werden auch in allen anderen Feldern der Streudiagramm-Matrix optisch und zeitgleich hervorgehoben. Verschiebt man beispielsweise den Rahmen entlang der niedrigen Werte einer bestimmbaren Variablen in einem ausgewählten Feld, wählt also kleine Intervalle der zweiten Variablen aus, so kann man das Aufleuchten der Daten zu den ausgewählten Beobachtungen auch bei den anderen Variablen verfolgen, und so, falls vorhanden, bedingt Abhängigkeiten aufdecken. Auf diese Weise werden nicht selten Zusammenhänge sichtbar, die anderweitig nicht zu entdecken waren. Im Bild 2 wurden 2 Punkte mit dem Rahmen ausgewählt (fette Punkte), bei denen es sich offensichtlich um Ausreißer handelt, da sie auch in anderen Streudiagrammen isoliert liegen.

Prinzipiell sind 4 dynamische Grundoperationen möglich:

das *Hervorheben von Daten* in einem Rahmen – eine Arbeitsweise zur Clustersuche. Manchmal ist es günstig, die Punkte nur solange hervorzuheben, solange sie in dem Rahmen sind und danach wieder zu inaktivieren. Das ist besonders bei umfangreichen Daten anzuwenden.

Durch das *Separieren von Punktclustern* oder Ausreißern lassen sich die verbleibenden Daten leichter nach Strukturen durchsuchen.

Schließlich sind das *Markieren* und *Identifizieren* von Punkten oder Punktclustern wichtige Grundoperation.

Durch die Kombination von rotierenden 3-D-Punktwolken mit den Möglichkeiten des Einrahmens lassen sich zusätzlich Einblicke in die multivariante Datenstruktur erhalten, dabei wurden das Instrumentarium und die Möglichkeiten dynamischer Grafik nur angedeutet /6/.

Hard- und Softwareanforderungen

Interaktive Datenanalyse und dynamische Grafik stellen hohe Anforderungen an die Hardware. Gegenwärtig sind dynamische Systeme meist noch an Workstations gebunden, die über wenigstens 1 MByte RAM, eine Leistung von mehr als 1 MFLOPS (Millionen Gleitkommaoperationen pro Sekunde), einen hochauflösenden Bildschirm (z. B. 1 024 × 1 024 Pixel) und ein grafisches Eingabesystem wie etwa die Maus verfügen.

Besonders geeignet sind Workstations mit Lisp-Architektur (Symbolics Lispmachine, Explorer, Macintosh II mit Micro-Explorer-Board). Hierbei kann der Bildschirm in mehrere Fenster aufgeteilt werden. Im Editor-Fenster wird das Lisp-Programm geschrieben, in mehreren Bild-Fenstern, die beliebig positioniert und in Größe und Form geändert werden können, laufen aktive grafische Prozesse gleichzeitig ab.

Unter dem Betriebssystem MS-DOS ist das Dynamic Graphics System DGS als Bestandteil des Programmsystems PC-ISP (Interactive Scientific Processor) verfügbar /7/. (Alle Bilder dieses Beitrages wurden mit diesem System erstellt.) Die Grundanforderungen hierfür sind 640 KByte RAM, ein CGA- oder EGA-/VGA-Graphikadapter und ein Farbmonitor (oder ein Schwarzweiß-Multisync-Monitor). Die qualitativen Unterschiede zwischen CGA und EGA werden in den Farbbildern 1 und 2 deutlich.

Das Interface-Konzept

Zur Steuerung von DGS werden Menüs verwendet. Die einzelnen Menüs (das Repräsentationsmenü auf dem Bildschirm oben, das Farbmenü unten und die überlagerten Menüs für Variablen, Teilmengen und Beobachtungen) werden durch Tastendruck ausgewählt, in den Menüs selbst bewegt man sich mit den Cursortasten oder der Maus.

Bei der Menüsteuerung werden nur wenige Bits an Informationen übertragen, so daß eine leichte Interaktion mit dem System möglich ist. Allerdings sind damit den Menüs auch Grenzen gesetzt, sie sind nicht für alle Dinge geeignet. Als Anwenderschnittstelle dient deshalb die matrixorientierte, interpretative Kommandosprache ISP (Interactive Scientific Processor), die – mit nahezu identischem Befehlsumfang der Apollo-Workstations – für Personalcomputer (als PC-ISP) beim Forschungsinstitut für Hygiene und Mikrobiologie Bad Elster erhältlich ist. PC-ISP ermöglicht eine flexible Dateneingabe und Datenausgabe, sowie die Dateiarbeit, umfaßt mächtige Matrix-Befehle, statische Standardberechnungen und weniger gebräuchliche statistische Operationen, aber auch Modellanpassung, Glättungsalgorithmen, Numerik, Zeitreihenanalyse und gestattet ein interaktives Experimentieren, ganz gleich, ob es sich um statistische oder andere Daten handelt. Farbbild 5 zeigt beispielsweise überlagerte Zeitreihen, wie sie mit ISP bearbeitet werden. Dabei ist ISP bemerkenswert schnell; bei Datensätzen mit einigen hundert Werten erscheint die Antwort auf die meisten Befehle augenblicklich, und die interaktive Arbeit bleibt auch bei Datensätzen durchführbar, die aus mehreren tausend Werten bestehen. Durch einfache Makroprogrammierung läßt sich der Befehlsumfang beliebig und individuell erweitern, und durch sogenannte Script-Files lassen sich ISP-Sitzungen beliebig nachvollziehen. Mit ISP als Anwenderschnittstelle werden wichtige, im allgemeinen zyklisch durchlaufene variierte Tätigkeiten des empirisch arbeitenden Datenanalytikers unterstützt. Fachwissenschaftler und Statistiker wählen versuchsweise einen analytischen Ansatz aus, führen dann einen Analyseschritt durch, betrachten die Daten auf dem Bildschirm (Übergang zu DGS), interpretieren und modifizieren das Zwischenergebnis nach gemeinsamer Beratung und führen den nächsten Schritt der experimentellen Informationsfindung aus.

Literatur

/1/ Tukey, J. W.: Exploratory Data Analysis. Addison-Wesley, Reading (Mass.) 1977

/2/ Fleischer, W.; Nagel, M.: Datenanalyse mit dem Personalcomputer. VEB Verlag Technik, Berlin (in Vorbereitung)

/3/ Henschke, K.; Nagel, M.: Die Exploration, Darstellung und Analyse von Daten – grafische Methoden der Datenanalyse. VEB Verlag Volk und Gesundheit, Berlin (in Vorbereitung)

/4/ Huber, P. J.: Statistical graphics: History and overview. Proc. Fourth Ann. Conf. and Exposition of the National Computer Graphics Assoc. National Computer Graphics Association, Fairfax (VA) (1983), S. 667

/5/ Fisherkeller, M. A.; Friedman, J. H.; Tukey, J. W.: PRIM 9: An interactive multidimensional data display and analysis system. Data: Its Use, Organization, and Management. Association for Computing Machinery, New York (1975), S. 140

/6/ Cleveland, W. S.; McGill, M. E. (Eds.): Dynamic Graphics for Statistics. Wadsworth & Brooks, Belmont (Cal.) 1988

/7/ PC-ISP Benutzerhandbuch mit Befehlsbeschreibung. Version 2.11. Artemis Systems, Inc., Carliste (USA) und Datavision, Klosters (Schweiz), 1985 und 1988

✉ KONTAKT ☏

Forschungsinstitut für Hygiene und Mikrobiologie Bad Elster, Heinrich-Heine-Str. 12, Bad Elster, 9933; Tel. 62 45

Verwaltung programmierbarer Zeichengeneratoren

Dr. Reiner Hirschmann
Hochschule für Verkehrswesen Dresden, Sektion Militärisches Transport- und Nachrichtenwesen

Einführung

Beim rechnergestützten Erzeugen von Bildern mittels Rasterdisplays setzen sich sowohl im monochromatischen als auch im polychromatischen Bereich in zunehmendem Maße sogenannte vollgrafische Geräte durch. Das ist in erster Linie durch die ständig sinkenden Kosten für die notwendigen großen Bildwiederholspeicher (Pixelspeicher) und den Einsatz immer leistungsfähigerer CPUs im Mehrprozessorverbund bedingt. Für solche vollgrafischen Rasterdisplays sind inzwischen umfangreiche und mit hohem Komfort versehene Softwarepakete entwickelt worden und weltweit im Einsatz /1/. Voraussetzung dafür sind allerdings Hostrechner mit großem Leistungsvermögen. Auch für 8-Bit-Bürocomputer existieren leistungsstarke Grafikprogramme, mit denen das Editieren eines Computerbildes relativ einfach und praktisch im Echtzeitbetrieb möglich ist.

Ein wichtiges Teilproblem stellt das Speichern fertiger Bilder auf externen Datenträgern dar. Bei komfortablen Systemen werden *Metafiles* verwendet, deren einzelne Objekte bei Aufruf durch die Systemsoftware das fertige Bild generieren. Durch 8-Bit-Mikrorechner kann mit diesem Verfahren kein Echtzeitaufruf fertiger Bilder erreicht werden. Für eine Reihe von Problemen sind aber das externe Speichern einer großen Zahl vorgefertigter Bilder mit geringem technischen Aufwand und der verzögerungsfreie Wiederaufruf von entscheidender Bedeutung (DIA-Prinzip bei Auskunftssystemen und im rechnergestützten Unterricht, programmiertes Lernen, Anzeigen in Industriesteuerungsanlagen u. ä.). Für die Lösung solcher Aufgaben bieten sich bei Mikrorechnern der unteren Leistungsklasse *pseudografische Verfahren* an (Bildwiederholspeicher als Bildelementcodespeicher in Zusammenarbeit mit einem Zeichengenerator). Zwar sind dann die Möglichkeiten beim Entwurf und bei der Konstruktion von Bildern etwas eingeschränkt, aber auf Grund des im allgemeinen um mindestens eine Zehnerpotenz niedrigeren Speicherbedarfs besitzen solche Geräte den großen Vorteil, daß sie fertige Bilder als topologische Strukturen abspeichern und damit eine minimale Generierungszeit bei Wiederaufruf gewährleisten können.

Um beim Editieren der Bilder nicht allzu stark eingeschränkt zu werden, verwenden leistungsstarke Rasterdisplays mit Bildelementcodespeicher oftmals programmierbare Zeichengeneratoren, deren Inhalt von Bild zu Bild gewechselt werden kann. Einige solcher Geräte ermöglichen bei schwach besetzten Bildern (Sparce-Grafik) das Erzeugen polychromatischer Grafiken, die nicht von denen vollgrafischer Geräte zu unterscheiden sind (Farbstruktur im Pixelraster) /2/. Zielstellung bei der Bildkonstruktion ist die Organisation eines möglichst hohen Wiederholgrades einzelner Bildelemente und die optimale Auslastung des Zeichengenerators. Dabei spielt das Geschick des Konstrukteurs eine wesentliche Rolle, wobei sich der notwendige Mehraufwand bei den angeführten Einsatzfällen meist lohnt.

Plazierung neuer Bildelemente

Völlig unabhängig von ihrem Aufbau können Bildwiederholspeicher (BWS) und programmierbare Zeichengeneratoren (ZG) als spezielle Datenstruktur vom Filetyp aufgefaßt werden. In solch einem Modell enthält jeder Datensatz des Bildwiederholspeichers einen Bildelementecode (BEC), wobei die üblicherweise ebenfalls vorhandenen Attributinformationen nicht beachtet werden. Jeder Datensatz des Zeichengenerators enthält die Strukturinformationen eines vollständigen Bildelementes (BE). Die Datensätze beider Files können von der CPU im Direktzugriff erreicht werden. Beim Bildrefresh ist der Zugriff der Steuerhardware auf den Bildwiederholspeicher rein sequentiell, auf den Zeichengenerator ebenfalls im Direktmodus.

Ein Objekt (Gerade, Kreis u. ä.) wird als Folge einzelner Bildelemente konstruiert. Nach Fertigstellung ist ein Bildelement an der richtigen Stelle im Gesamtbild zu plazieren, wobei jedem Bildelement-Platz eindeutig eine feste Datensatznummer des Bildwiederholspeichers zugeordnet ist.

Mit den Bezeichnungen

– BWS (j) oder ZG (i): Inhalt der Datensätze j bzw. i

– k: Anzahl der BE im Gesamtbild (Größe der Datei BWS)

– n: Anzahl der möglichen BEC (Größe der Datei ZG),

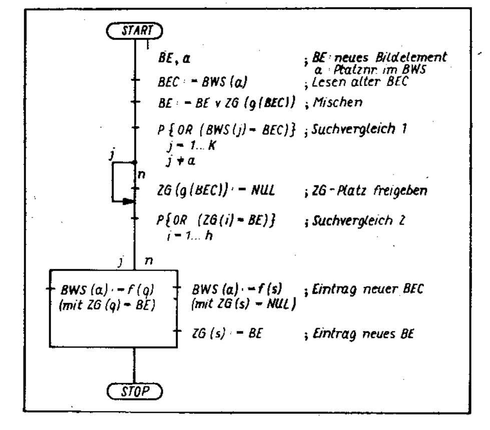


Bild 1 Plazierung eines neuen Bildelementes im Gesamtbild

und der eindeutigen Zuordnung BEC = f (i) bzw. i = g (BEC) gilt der Algorithmus nach Bild 1.

Zunächst wird der an dieser Stelle bisher plazierte Bildelementecode gelesen und das ihm zugeordnete Bildelement mit dem neuen Bildelement zu einem komplexen Bildelement gemischt. Danach erfolgt eine Prüfung, ob das bisherige Bildelement einmalig im Gesamtbild vorhanden war (Suchvergleich 1). Wenn ja, wird der entsprechende Datensatz im Zeichengenerator freigegeben. Damit wird gewährleistet, daß nur benötigte Bildelemente Platz im Zeichengenerator beanspruchen. Im weiteren wird geprüft, ob das komplexe neue Bildelement selbst schon im Zeichengenerator vorhanden ist (Suchvergleich 2), das heißt ob dafür schon ein Bildelementecode existiert. Wenn ja, erfolgt dessen Eintragung an vorgesehener Stelle im Bildwiederholspeicher (damit erhöht sich der Wiederholungsgrad dieses Bildelementecodes um 1); wenn nein, so wird ein leerer Datensatz im Zeichengenerator gesucht, dort das komplexe Bildelement eingetragen und der zugeordnete Bildelementecode im Bildwiederholspeicher plaziert. Wird kein Leersatz im Zeichengenerator gefunden, ist die Kapazität des Gerätes erschöpft (nicht gezeichnet).

Als besonders hemmend für einen schnellen Bildaufbau erweisen sich dabei die beiden zeitintensiven sequentiellen Suchvergleiche. Wenn in den Datensätzen des Zeichengenerators neben den Informationsbits der Struktur der Bildelemente weitere Bits zur Kennzeichnung des Wiederholgrades des betreffenden Bildelementes reserviert und verwaltet werden, kann der Suchvergleich 1 entfallen. Das ist zum Beispiel ohne erhöhten Aufwand manchmal dann möglich, wenn die Matrixstruktur des Bildelementes nicht durch ein 2^n-Byteraster dargestellt werden kann und damit in den Datensätzen des Zeichengenerators redundante Bits vorhanden sind.

Bild 2a zeigt aber, daß die ausschließliche Organisation aller Suchoperationen unter Verwendung von Datenstrukturen, die unmittelbar am Bildrefresh beteiligt sind, nicht zweckmäßig sind.

Nur durch entsprechende Synchronisation (z. B. CPU-Zugriff im Bildrücklauf) ist ein störungsarmer Bildaufbau möglich, allerdings auf Kosten eines etwa 5fachen Zeitbedarfs

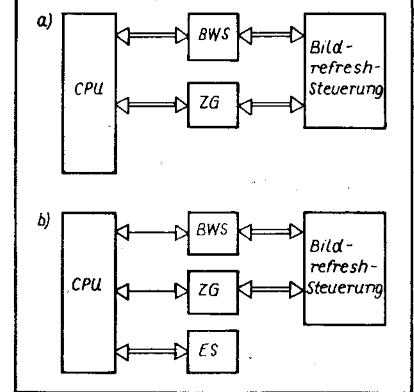


Bild 2 Datenfluß (qualitativ) beim Bildaufbau und beim Bildrefresh ohne (a) und mit (b) Ergänzungsstruktur ES
↔ geringer Datenfluß im Zeitmittel
⟺ großer Datenfluß im Zeitmittel

für den Konstruktionsablauf. Eine günstigere Variante stellt die Verwendung einer dritten, nur im CPU-Zugriff stehenden Ergänzungsstruktur (ES) dar (Bild 2b).

Beispielsweise wäre das vollständige Doublieren von Bildwiederholspeicher und Zeichengenerator durch Ergänzungsstrukturen denkbar. Dann verbleiben als CPU-Zugriffe zum Bildwiederholspeicher und zum Zeichengenerator nur die Einschreibaktionen;

Bildstörungen sind auch im unsynchronisierten Fall kaum wahrnehmbar. Nachteilig ist der relativ große Speicherbedarf (5–7 KByte), so daß diese Variante insbesondere beim Einsatz speicherintensiver Sprachen und Interpreter (Turbo-Pascal, Basic) im Rahmen der 8-Bit-Standardtechnik unzweckmäßig ist.

Verwendung von Suchcodes

Das nachfolgend beschriebene Verfahren zeigt eine Möglichkeit, wie durch die spezielle Ergänzungsstruktur ZGMAP eine zeit- und speichereffektive Verwaltung der redundanzlosen Strukturen Bildwiederholspeicher und Zeichengenerator unter Gewährleistung eines geringen Störgrades beim Bildaufbau organisiert werden kann. ZGMAP umfaßt gleichviel Datensätze wie der Zeichengenerator, dabei enthält ZGMAP(i) die ZG(i) zugeordneten Informationen. Die Datensatzstruktur von ZGMAP besteht aus zwei Feldern. Feld 1 beinhaltet Informationen über den Wiederholgrad des zugeordneten Bildelementes im Gesamtbild (0 = Leersatz des ZG). Feld 2 dient als Schlüsselfeld für den Suchvergleich 2. Die Schlüsselinformation wird durch geeignete Verdichtung aus der Informationsstruktur jedes erzeugten Bildelementes gebildet. Ziel ist die eindeutige Abbildung der „langen" Informationsstruktur (typisch 8 oder 16 Byte) auf einen „kurzen" Suchcode (typisch 1 Byte). Obwohl eine solche Abbildung wegen des Informationsgehaltes der Bildelemente-Strukturen nicht umkehrbar eindeutig sein kann, ist es möglich, durch geeignete Abbildungsfunktionen eine relativ gleichmäßige Verteilungsdichte für die Suchschlüssel aller Bildelemente, die zu einem Bild gehören, zu gewährleisten. Damit kommen gleiche Suchschlüssel für verschiedene Bildelemente nur mit geringer Wahrscheinlichkeit in ZGMAP vor. Die Verwaltung von Bildwiederholspeichern und Zeichengeneratoren mit ZGMAP zeigt Bild 3.
Der Suchvergleich 1 reduziert sich auf das Rückwärtszählen der entsprechenden ZGMAP-Information (Feld 1), wobei gegebenenfalls automatisch die Freigabe des zugeordneten Datensatzes des Zeichengenerators erfolgt. In diesem Falle wird auch der ausschließlich Leersätze kennzeichnende Suchcode 0 in Feld 2 eingetragen, um diese Leersätze vom Suchvergleich 2 auszuschließen.
Der Suchvergleich 2 erfolgt in zwei Stufen. Im sequentiellen Zugriff werden die Datensätze von ZGMAP durchsucht, bei übereinstimmendem Suchschlüssel erfolgt der Direktzugriff zum Zeichengenerator und der Vergleich der vollständigen Bildelemente-Strukturinformation. Der weitere Ablauf entspricht dem in Bild 1 mit den für die Aktualisierung von ZGMAP notwendigen Erweiterungen.

Realisierungsvariante

Beim Mustergerät /2/ sind Bildwiederholspeicher und Zeichengenerator redundanzlose Hardwarestrukturen mit den Kennwerten k = 1 024 (Platzzahl im BWS), n = 192 (mögliche BEC) und 16 × 8 Bit Punktraster der Bildelemente. ZGMAP ist softwaremäßig als Array-Struktur mit insgesamt 192 × 2 Byte realisiert.
Ziel der Implementierung war geringer Speicherbedarf bei schnellstmöglichem Zugriff. Als Suchcode wurde ein CRC-Restpolynom von 1 Byte verwendet, wobei der Wert 0 ausgeschlossen ist. Damit kann eine gute Gleichverteilung aller Suchcodes eines realisierten Bildes erreicht werden. Um die mittlere Auslastung von Feld 1 (ZGMAP) nicht allzu uneffektiv werden zu lassen und auch um bei dessen Informationsverarbeitung schnelle 8-Bit-Operationen benutzen zu können, wurde auf eine absolute Zählung des Wiederholungsgrades (maximal 1 024/BE) verzichtet. Nach Erreichen eines maximalen Zählwertes (128) wird auch bei erneutem Einschreiben des zugehörigen Bildelementecodes in den Bildwiederholspeicher keine weitere Erhöhung vorgenommen. Ergibt sich im Verlaufe der Bildkonstruktion die Löschung eines solchen Bildelementecodes im Bildwiederholspeicher, so wird beim notwendigen Rückwärtszählen von ZGMAP ein Markierungsbit gesetzt. Wenn insgesamt 128 gleiche Bildelementecodes wieder aus dem Bildwiederholspeicher entfernt worden sind, kann aus der ZGMAP-Information keine Aussage mehr über den Wiederholgrad dieses Bildelementecodes getroffen werden. Deshalb wird zu diesem Zeitpunkt eine spezielle Zählprozedur wirksam, die den ganzen Bildwiederholspeicher nach diesem Bildelementecode durchsucht und das Ergebnis in Feld 1 einträgt. Da entsprechend der Wahrscheinlichkeit für das Auftreten gleicher Bildelementecodes dieser Fall nur sehr selten vorkommt, tritt im Mittel keine merkliche Zeitverzögerung oder Bildstörung auf.

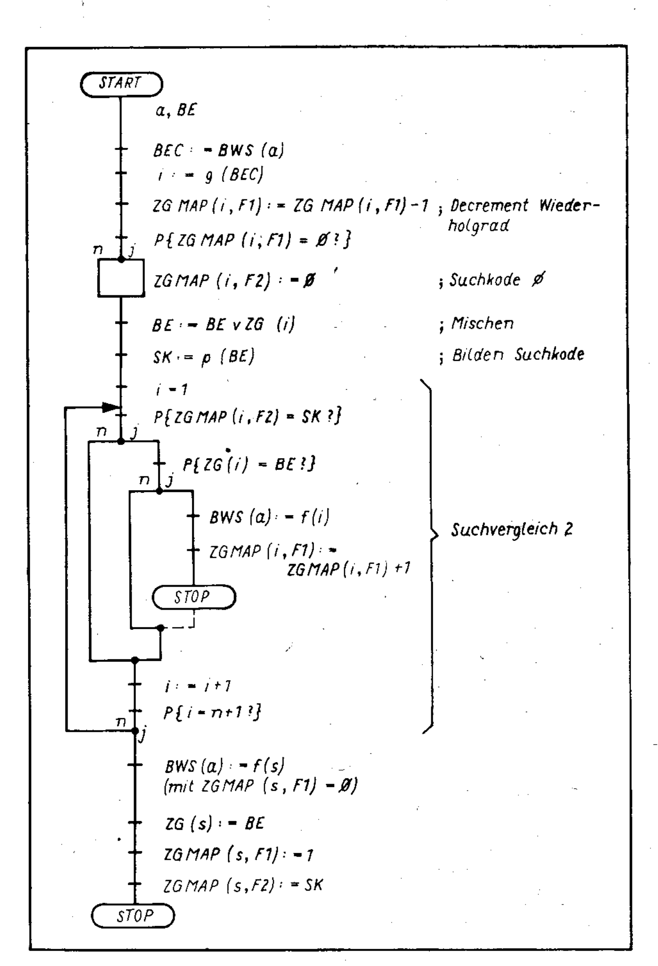


Bild 3 Plazierung eines neuen Bildelementes unter Verwendung von ZGMAP

Praktische Ergebnisse

Die besprochene Verwaltungsorganisation wurde in ein Paket grafischer Grundfunktionen implementiert, das seit rund vier Jahren erfolgreich zur Ausgabe von Farbgrafiken hoher Qualität durch Bürocomputer BC 5120 genutzt wird. Die Universalität des Verfahrens wird durch die verwendeten Betriebssysteme (SCP, UDOS, WEGA) und Programmiersprachen (ASM, BASI, Turbo-Pascal, Basic, PLZSYS, C) demonstriert. Der Gesamtspeicherbedarf für die Verwaltung von Bildwiederholspeicher und Zeichengenerator beträgt etwa 900 Byte. Der unsynchronisierte CPU-Zugriff gewährleistet einen schnellen Bildaufbau. Die verbleibenden Reststörungen werden von den meisten Anwendern nicht qualitätsmindernd gewertet. Im Vergleich mit anderen untersuchten Verwaltungsformen (synchronisierter und unsynchronisierter Zugriff mit einfacher Belegungsliste des Zeichengenerators, softwaremäßiges Puffern von Bildwiederholspeicher und Zeichengenerator) stellt das Verfahren einen guten Kompromiß zwischen den Bewertungskriterien Speicherbedarf und Störungen beim Bildaufbau dar.

Literatur

/1/ Kotzauer, A.: Standardisierte Grafik-Software. edv-aspekte, Berlin 4 (1985) 2, S. 18

/2/ Hirschmann, R.: Technische Beschreibung CD-SMTN, Anwenderdokumentation. Hochschule für Verkehrswesen Dresden, Sektion MTN

/3/ Schreiber, M.: Grafische Bildschirmansteuerungen für Kleincomputer. Mikroprozessortechnik, Berlin 1 (1987) 8, S. 249

✉ KONTAKT ☎

Hochschule für Verkehrswesen „Friedrich List", Sektion MTN, Friedrich-List-Platz 1, Dresden, 8010; Tel. 4 62 68 69

Echtzeituhr für Mikrorechner

Konrad Malsch
VEB Mikroelektronik
„Wilhelm Pieck" Mühlhausen

Einführung

Die Echtzeituhr (EZU) generiert elektronisch die aktuelle Uhrzeit unabhängig von eventuellen Netzspannungsausfällen oder -abschaltungen des übergeordneten Rechners. Fehlfunktionen durch gewollt oder ungewollt falsche Zeiteingaben, wie sie bei herkömmlichen „Softwareuhren" möglich sind, entfallen somit. Gegenüber Funkuhren besitzt die EZU den Vorteil, daß die Zeitinformation binnen weniger Millisekunden nach Spannungszuschaltung zur Verfügung steht.

Das vorliegende Konzept wurde speziell für Mikrorechner auf der Basis des Prozessors U 880 entwickelt. Durch geringfügige Hard- und Softwaremodifikationen läßt es sich an spezifische Systembedingungen problemlos anpassen. Auch der Einsatz im Rahmen des 16-Bit-Mikrorechnersystems U 8000 ist denkbar. Besonders günstig gestaltet sich die Zusammenschaltung mit Rechnern, die über einen batteriegestützten CMOS-RAM verfügen, oder mit Einchipmikrorechnern, die im Power-down-Modus betrieben werden können. In einem solchen Fall beschränkt sich der Stellvorgang statt der sonst erforderlichen Tastatursimulation auf das Ablegen eines Korrekturwertes im batteriegestützten Speicher. Minimal zwei Eingabeleitungen reichen so zur Übernahme der Uhrzeit aus, welche nach Verknüpfung mit dem oben genannten Korrekturwert die Echtzeit ergibt.

Technische Daten
- Bereitstellung von Wochentag, Tag, Stunden, Minuten, Sekunden im direkten BCD-Code (gepackt)
- 5 freie Portleitungen (Bitmodus) für Steuerzwecke
- Einplatinenbaugruppe 115 mm × 122,5 mm
- Funktionsdauer mit einem Batteriesatz > 1 Jahr
- Genauigkeit: im Mittel ± 20 Sekunden/Monat im Umgebungstemperaturbereich von 20 bis 25 Grad Celsius
- Stromversorgung: +5 V, ca. 0,3 A; −5 V, ca. 0,02 A; 2 Stück Silberoxidknopfzellen oder 1 Stück Lithiumzelle CR 2032

Funktionsweise

Das Funktionsprinzip der Schaltung wird im folgenden anhand des Blockschaltbildes (Bild 1) erläutert. Kernstück der Schaltung ist der quarzgenaue Zeitgeber auf der Basis des Uhrenrechnerschaltkreises U 825 G (12-Stunden-Zähler) oder U 827 G (24-Stunden-Zähler). Dieses Bauelement zeichnet sich gegenüber vergleichbaren CMOS-Uhrenschaltkreisen (z. B. U 131 G) dadurch aus, daß das Anzeigeregister, das nach dem Einschalten die Uhrzeit enthält, zu Testzwecken ständig seriell ausgegeben wird /2/. Eine Decodierung der LCD-Ansteuersignale wäre auf Grund des erheblichen Aufwandes nicht vertretbar.

Über ein paralleles Ein-/Ausgabeinterface (PIO) werden die serialisierten Zeitinformationen in das Anwendersystem eingelesen sowie Stellinformationen an den Zeitgeber übermittelt. Zur Kopplung dieser Baugruppen sind Pegelanpaßstufen sowie Komparatoren erforderlich. Außerdem muß eine geeignete Einordnung der PIO in das Mikrorechnersystem mittels Adreßdecoders und Steuerlogik erfolgen. Der computerseitige Anschluß der EZU wird über die in Mikrorechnern allgemein üblichen Bussysteme gebildet. Im vorliegenden Anwendungsfall macht es sich erforderlich, daß Adreß- sowie Steuersignalbus getrieben werden, während der Datenbus ungepuffert verwendet wird.

Durch die Steuerlogik werden folgende Funktionen erfüllt:
- Aussenden eines Strukturbytes auf Anforderung durch den Computer
- Speichern und Anzeige der Betriebsart (online/offline)
- systemgerechte Einbindung der EZU in die Prioritätsketten
- Verzögerung des Steuersignals /IORQ

Zur Senkung des Schaltungsaufwandes kann unter Umständen der Umfang dieser Logikbaugruppe auf ein Minimum (Realisierung der Prioritätsketten) reduziert werden. Der Adreßdecoder bildet aus der Adresse 078 H sowie zwei Freigabesignalen der Steuerlogik das Signal /CS für die PIO. Somit wird der Adreßbereich 078 H bis 07 BH belegt. Eine Anpassung an andere E/A-Adressen ist durch eine entsprechende Beschaltung des Decoders leicht vorzunehmen. Der PIO-Schaltkreis ist systemseitig mit dem ungetriebenen Datenbus und den erforderlichen Steuersignalen verbunden. Da ein spezieller Rücksetzanschluß fehlt, wird diese Funktion durch UND-Verknüpfung der Signale /M1 und /RESET ermöglicht /3/.

Tor A der PIO ist mit den Tastaturanschlüssen K1–K8 des Uhrenrechnerschaltkreises verbunden. Über diese Leitungen werden Tastatureingaben softwaremäßig simuliert (siehe Programmbeschreibung). Außerdem liefert die Leitung K7 das Synchronsignal zur seriellen Datenschnittstelle am U 825 G. Für die genannten Funktionen werden die Anschlüsse teilweise als Ein- *und* Ausgänge benutzt.

Tor B übernimmt auf je einer Leitung die seriellen Daten, ein Batteriezustandssignal sowie Sekundenimpulse, aus denen Interrupts generiert werden. Die restlichen 5 Toranschlüsse sind unbeschaltet und können im Bitmodus anderweitig verwendet werden.

Die Pegelanpassung zwischen PIO und Zeitgeberschaltkreis ($U_B = 3$ V) ist nur für PIO-Ausgänge erforderlich und erfolgt über Spannungsteiler. Sie garantieren, daß die maximale Eingangsspannung für den U 825 G (zirka 3,3 V) nicht überschritten wird. Im Ruhezustand werden die PIO-Anschlüsse auf Eingabe programmiert, so daß die Tastaturanschlüsse unbeeinflußt bleiben.

Der Zeitgeberschaltkreis wird, was Takt und Stromversorgung betrifft, in seiner Standardbeschaltung eingesetzt. Von den LC-Display-Anschlüssen findet lediglich Leitung B3, die das im Sekundentakt blinkende Minuszeichen zwischen Stunden- und Minutenanzeige ansteuert, Verwendung. Der zugehörige Komparator steuert ein nachtriggerbares Monoflop an, wenn die Spannung am invertierenden Eingang größer als 2,3 V ist (Segment B3 angesteuert). Die Haltezeit des Monoflops wurde so gewählt, daß die Multiplexfrequenz der Anzeige am Ausgang nicht in Erscheinung tritt /2/. Somit steht eine Frequenz von 1 Hz mit einem Tastverhältnis von 0,5 zur Gewinnung von Sekundeninterrupts zur Verfügung. In der Betriebsart *online* gelangen die Sekundenimpulse zum Anschluß 1 des PIO-Tors B und takten gleichzeitig eine Zustandsanzeige.

Die Betriebsspannung des U 825 G wird mit Hilfe eines Komparators auf ihren Minimalwert von 2,8 V getestet. Einer entsprechenden Programmroutine obliegt die Bewertung des bei zu niedriger Batteriespannung entstehenden L-Pegels am Eingang B2 der PIO.

Am Anschluß TS3 des U 825 G (Open-Drain-Ausgang) wird der serielle Datenstrom hochohmig abgegriffen und über einen Komparator auf TTL-Pegel umgesetzt. In allen Komparatoren kommen BIFET-Operationsverstärker zum Einsatz. Dadurch konnte die Dauerbelastung der Uhrenrechnerstromversorgung gering (zirka 0,015 mA) gehalten werden. Zur Anpassung der Operationsverstärkerausgänge an die TTL-Bedingungen der PIO-Eingänge dienen Dioden-Widerstandskombinationen.

Programmbeschreibung

Die Software zur EZU umfaßt eine Initialisierungsroutine, eine Interruptroutine zum Auslesen der Zeitinformation aus dem U 825 G sowie ein Programm zum Stellen der Uhr. Im Rahmen der Initialisierung wird die EZU *online* geschaltet, sowie die Batteriespannung abgefragt. Aus diesen beiden Aktivitäten können Fehlermeldungen resultieren. Weiterhin werden die Portleitungen der PIO als Ein- oder Ausgänge festgelegt, die Ports in den Bitmodus geschaltet sowie Leitung B1 zur Interruptauslösung (H-L-Flanke des Sekundentaktes) freigegeben. Nach Bereitstellung des Interruptvektors und Aktivierung der Interruptlogik der CPU (EI) wird der erste Se-

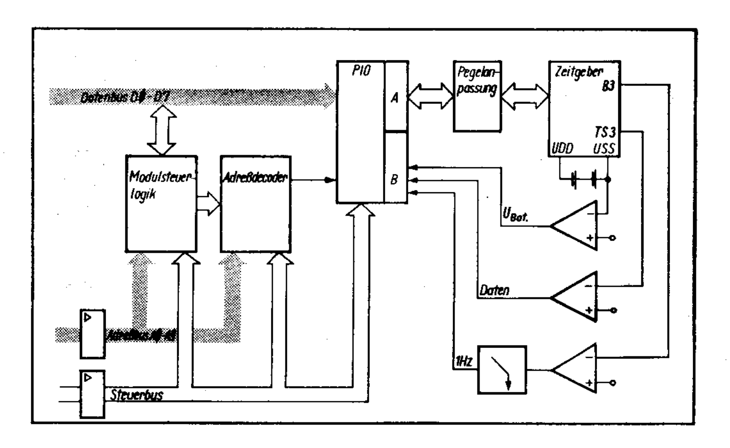


Bild 1 Blockschaltbild der Echtzeituhr

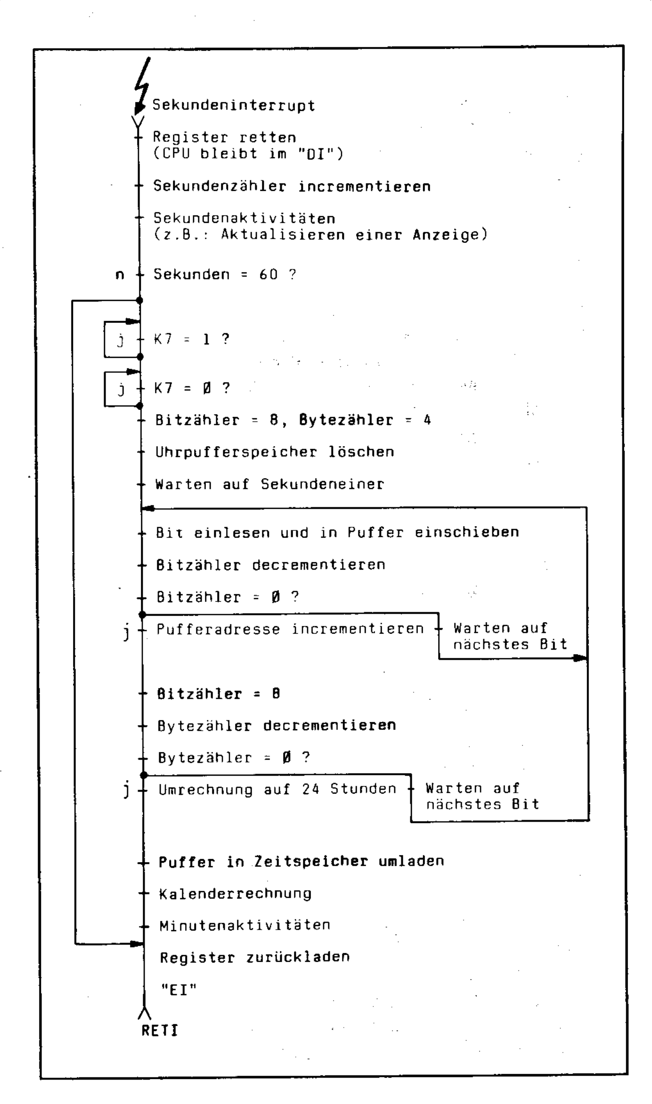


Bild 2 PAP zum Auslesen der Uhrzeit aus dem U 825 G im Minutentakt

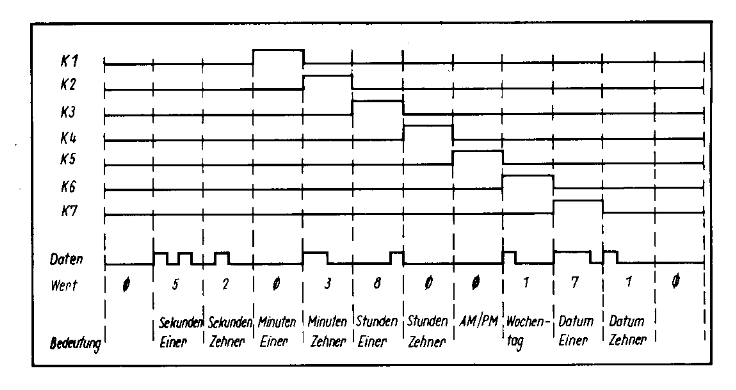


Bild 3 Zuordnung der Zeitinformation zu den K-Impulsen (1 Zyklus)

kundeninterrupt abgewartet, so daß beim Verlassen der Initialisierungsroutine nach maximal einer Sekunde bereits die aktuelle Uhrzeit im Speicher verfügbar ist.

Der Programmablauf zur Gewinnung der Uhrzeit ist in Bild 2 dargestellt. Außerdem wird in Bild 3 die Zuordnung der Zeitinformation (verdeutlicht am Beispiel 8.30 Uhr) zu den Tastaturimpulsen ersichtlich. In der Zeiterfassungsroutine werden die Sekunden gezählt, zu jeder vollen Minute die Zeit (Sekunden, Minuten, Stunden, Wochentag) ermittelt und entsprechende Umrechnungen (24-Stunden-Anzeige, Kalender) vorgenommen. Die Synchronisation für den seriellen Datenaustausch bildet der Tastaturimpuls K7. Wurde dieser im Polling-Modus erkannt, so kann nach Durchlaufen einer Warteschleife (16,5 Bittakte = 33 Systemtakte des U 825 G) das erste Bit der Sekundeneiner eingelesen werden. Die weitere Bitsynchronisation erfolgt über Warteschleifen, wobei zur genauen Einhaltung des Bittaktes andere Systeminterrupts gesperrt werden (DI). In diesem Programmteil, der aus Zeitgründen (bei Rechnertaktfrequenzen um 2 MHz) nicht völlig interruptgesteuert ablaufen kann, ist eine genaue Abstimmung auf die Taktfrequenz des verwendeten Mikrorechners vorzunehmen. Dazu können für den logischen Programmablauf bedeutungslose Befehle (NOP; LD A,A; ...) eingefügt werden. Das vorliegende Programm wurde an eine Taktfrequenz von zirka 1,75 MHz angepaßt.

Da im Datumsspeicher des U 825 G nur 31 Tage gezählt werden, wird auf das Auslesen dieser Information verzichtet und die Kalenderrechnung einschließlich der Verarbeitung der Schaltjahre bei Erkennen der Uhrzeit 0.00 Uhr durchgeführt. Für Tastatureingaben zum Stellen der Uhrenregister des U 825 G mittels eines Mikrorechners bieten sich verschiedene Nachbildungsmöglichkeiten an /2/. Gewählt wurde die Variante mit dem geringsten Hardwareaufwand, indem die benötigten Tastaturanschlüsse (K1–K8) mit PIO-Anschlüssen (Tor A) verbunden werden. Das zugehörige Programm erfüllt zur Nachbildung von Tastenbetätigungen (elektrische Verbindung zweier Tastaturanschlüsse) folgende Funktionen:

– Abfragen des zur Taste gehörigen Ausganges auf H
– Anlegen von H an den zugehörigen Eingang, indem die entsprechende PIO-Leitung auf Ausgabe programmiert und ein H ausgegeben wird
– Abschalten dieses H-Impulses, wenn der Tastaturausgang auf L geht (PIO-Leitung wieder auf Eingabe schalten)

Dieser Ablauf wird 15mal wiederholt, um ein sicheres Erkennen der „Taste" durch den U 825 G zu gewährleisten. Danach folgt noch eine Warteschleife zum Überbrücken der Zifferneingabezeit. Das Programm zur Ausgabe der Stellinformationen trennt gepackte BCD-Ziffern, testet diese auf Pseudotetraden und übergibt an das eigentliche Tastatur-Unterprogramm Steuerinformationen über die jeweils benötigten Ein- und Ausgabeleitungen. Das gesamte Programm umfaßt etwa 650 Byte Maschinencode.

Literatur

/1/ Malsch, K.: Echtzeituhr für Kleincomputer. Nachnutzungsdokumentation, VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, 1987
/2/ Kirves, K. D.; Bendlin, G.: Einatzmöglichkeiten U 825 G und U 824/26 G – Studie. VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, 1983
/3/ Kieser, H.; Meder, M.: Mikroprozessortechnik. Berlin: VEB Verlag Technik 1982

⊠ KONTAKT ☏

VEB Mikroelektronik „Wilhelm Pieck" Mühlhausen, Abt. Technologische Projektierung, Eisenacher Straße 40, Mühlhausen, 5700; Tel. 58 73 63

TERMINE

Fachtagung „Anwendererfahrungen und Ausblicke auf die Weiterentwicklung des Programmsystems MIDOS für die Nutzung in der wissenschaftlich-technischen Information und im Bibliothekswesen"

WER? Bezirksvorstand Berlin der KDT und Betriebssektion der KDT des VEB Elektro-Apparate-Werke Berlin-Treptow „Friedrich Ebert"
WANN? 1. November 1989
WO? Berlin, Museum für Deutsche Geschichte, Kinosaal, Unter den Linden 2, Berlin, 1086
WAS?
- Entwicklungsstand und Ausblick
- Einsatz von MIDOS im dokumentographischen Informationsrecherchesystem AGROSELEKT
- Anwendung von MIDOS 16 für das Dokumentrecherchesystem des Binnenhandels, Herstellung von Bibliographien
- Datenaustausch mit dem Mikrorechner MITRANS
- Erfahrungen mit MIDOS 8 in der Dieselmotorenforschung
- Erfahrungen mit MIDOS im Verkehrswesen bei der Verwendung unterschiedlicher Rechnersysteme

WIE? Interessenten wenden sich bitte an: Kammer der Technik, Bezirksverband Berlin, Bereich Wissenschaft und Technik, Kronenstraße 18, Berlin, 1080; Tel. 2 00 03 61, App. 75

Lange

Heap-Nutzung unter Turbo-Pascal auf 16-Bit-Rechnern

Bernd Matzke

Die Programmiersprache Turbo-Pascal hat sich dank ihrer vielen Vorteile unter den auf PCs verfügbaren Sprachen einen Spitzenplatz erobert.
Beim Übergang auf 16-Bit-Systeme tritt der Nachteil der Beschränkung von Programm- und Datenspeicherbereich auf maximal 64 KByte nachteilig in Erscheinung. Der vorhandene, wesentlich größere Hauptspeicher kann nicht direkt für statisch definierte Variablen genutzt werden, sondern steht nur für dynamische Variablen zur Verfügung. Da auf diese nur durch Verwendung von Zeigern zugegriffen werden kann, werden sie wegen mangelnder Kenntnisse oder vermeintlicher umständlicher Handhabung selten verwendet, der vorhandene Hauptspeicher wird nicht genutzt. Umfangreiche Datenbestände werden auf ein externes Speichermedium ausgelagert und die damit verbundenen Nachteile in Kauf genommen. Aber gerade auf 16-Bit-Systemen kann durch die Nutzung des Heap eine erhebliche Geschwindigkeitssteigerung bei der Verarbeitung größerer Datenbestände erreicht werden. Im folgenden sollen drei praxisorientierte Möglichkeiten der Heap-Nutzung vorgestellt werden.

Daten-Cache

Dieses Verfahren dient zur Beschleunigung der Suche von Datensätzen aus sehr großen Diskettendateien. Es verwendet die meist zur Erläuterung des Zeiger-Konzepts verwendete verkettete Liste. Der Grundgedanke ist folgender:
Im Heap wird eine lineare Liste angelegt. Ein Datensatz wird jetzt zuerst in dieser Liste gesucht. Nur wenn dort der Datensatz nicht gefunden wird, erfolgt ein Zugriff auf die Diskette oder die Festplatte.
Der gefundene Eintrag (egal, ob von Diskette gelesen oder im Heap gefunden) wird an den Anfang der Liste gesetzt. Die Länge der Liste ist jedoch auf eine bestimmte Satzzahl begrenzt, bei deren Überschreitung der jeweils letzte Eintrag verlorengeht. Bei einer ausreichend großen Anzahl von Suchvorgängen werden so oft benötigte Einträge am Anfang der Liste konzentriert, seltener gesuchte befinden sich in der Liste weiter hinten oder nur auf der Diskette. Die Anzahl der Diskettenzugriffe und damit die durchschnittlich zur Suche eines Datensatzes benötigte Zeit verringert sich. Dieser Effekt ist um so größer, je häufiger bestimmte Datensätze gesucht werden und um so länger mit dem Programm gearbeitet wird.
Das Prinzip erinnert an das aus dem Bereich der Hardware bekannte Cache-Konzept.
Dieses Verfahren ist dann sinnvoll anwendbar, wenn auf die einzelnen Datensätze einer großen Datei mit unterschiedlicher Häufigkeit zugegriffen wird und diese Datensätze nur gelesen werden sollen.
Durch Verwendung dieses Verfahrens zum Suchen von Datensätzen mit Materialangaben konnte die Laufzeit eines Programms zum Schreiben von Rechnungen halbiert werden.

Die Länge der Liste ist so zu wählen, daß das lineare Suchen in der Liste wesentlich weniger Zeit erfordert als das Suchen auf der Diskette.
Im Programm heap1 sind nur die zum Anlegen und Aktualisieren der Liste notwendigen Prozeduren enthalten. Das Hauptprogramm und die Prozedur zum Suchen auf Diskette sind anwenderspezifisch und sollen hier nicht aufgeführt werden. Unter Umständen kann auch auf der 8-Bit-Technik trotz wesentlich geringerer Heap-Größen mit diesem Verfahren ein spürbarer Zeitgewinn erzielt werden.

Array mit Zeigern

Für kleinere Dateien, die vollständig im Heap Platz haben, ist das oben beschriebene Verfahren nicht unbedingt sinnvoll, da die lineare Suche in einer umfangreichen Liste mehr Zeit erfordern kann als ein optimal gestaltetes Suchverfahren zur Suche auf der Diskette.
In solchen Fällen kann der Heap durch die im Programm heap2 enthaltenen Prozeduren ähnlich wie eine RAM-Disk genutzt werden. Dabei werden die einzelnen Datensätze im

```
program heap1;
const freemem     = 1000;                  {je nach gewuenschter groesse der liste}
      fehlstring  = '@@@@@@@@@@';

type  zeiger      = ^et;
      et          = record
                      matnr:string[10];    {datensatzbeschreibung anwenderspez.}
                      preis:real;
                      next:zeiger;                         {zeiger obligatorisch}
                      last:zeiger;
                    end;

var erster_eintrag,letzter_eintrag:zeiger;                 {zeiger fuer liste}
    eintrag:et;

procedure suchen(var material:et);
{suchen des Satzes mit material.matnr und rueckgabe des vervollstaendigten }
{datensatzes, suchbedingung ist anwenderspezifisch}
var sucheintrag:zeiger;

  procedure fuellen(e:et;var z:zeiger);
  begin
    new(z);
    z^:=e;
  end;

  procedure vornansetzen(var z:zeiger);
  {dynamische Variable auf die der zeiger zeigt,an den anfang der liste setzen}
  begin
    if erster_eintrag = nil then
    begin
      erster_eintrag:=z;          {wenn allererste eintrag, dann Anfang erzeugen}
      erster_eintrag^.next:=nil;
      erster_eintrag^.last:=nil;
      letzter_eintrag:=erster_eintrag;
    end
    else
    begin
      z^.next:=erster_eintrag;
      z^.last:=nil;
      erster_eintrag^.last:=z;
      erster_eintrag:=z;
    end;
  end;

  procedure herausnehmen(var z:zeiger);
  {aus liste die variable z herausnehmen und die beiden nachbarn miteinander }
  {verketten, nicht auf ersten eintrag der liste anwenden}
  begin
    z^.last^.next:=z^.next
    if z^.next<> nil then z^.next^.last:=z^.last        {wenn z nachfolger hat}
                     else letzter_eintrag:=z^.last;     {wenn kein nachfolger }
  end;


procedure nach_vorne(var z:zeiger);
{verschieben einer variablen in dynamischer liste an den anfang}
begin
  if z <> erster_eintrag then
  begin
    herausnehmen(z);
    vornansetzen(z);
  end;
end;

procedure einsortieren(eintrag:et);{einfuegen des satzes in dynamische liste}
var temp_eintrag:zeiger;
begin
  if memavail > freemem then      {wenn genuegend speicherplatz dann anfuegen}
  begin
    fuellen(eintrag,temp_eintrag);          {neue dynamische variable erzeugen}
    vornansetzen(temp_eintrag);                {und an anfang der liste setzen}
  end
  else
  begin
    temp_eintrag:=letzter_eintrag;          {wenn nicht genuegend Speicherplatz, }
    herausnehmen(letzter_eintrag);          {dann letzten Eintrag ueberschreiben}
    temp_eintrag^:=eintrag;                              { und nach vorn schieben}
    vornansetzen(temp_eintrag);
  end;
end;
```

Heap als dynamische Variablen abgelegt und die Adresse der Variablen in einem Array abgespeichert. Dieses Array kann als statische Variable im Datensegment oder auch mittels Zeiger im Heap vereinbart werden. Im Beispiel wurde die letzte Variante verwendet.
Im Beispielprogramm werden 5 Prozeduren vorgestellt, die je nach Größe von Heap und Datei den Zugriff auf die Datei steuern. Sie entsprechen in ihrer Funktion den gleichnamigen Standardfunktionen.
Steht im Heap genügend Platz zur Verfügung, dann wird die komplette Datei in den Heap eingelesen. Alle Lese- und Schreiboperationen werden dann im Heap durchgeführt. Ein Rückschreiben der im Heap gespeicherten Daten erfolgt nur, wenn sie geändert wurden.
Ist der Heap zur Aufnahme der Datei zu klein, so wird automatisch die Diskettendatei als Eingabe- und Ausgabemedium genutzt.
Anwenderprogramme werden damit unabhängig von der Heap-Größe. Steht genügend Speicherplatz zur Verfügung, ergibt sich eine Beschleunigung des Programmablaufes, ansonsten als einzige negative Auswirkung lediglich ein größerer Zeitbedarf.
Der gleiche Effekt wäre natürlich auch durch die Verwendung einer RAM-Disk zu erzielen, die aber nicht unter allen Betriebssystemen und Rechnertypen zur Verfügung steht und die auch nicht durch Kommandos vom Anwenderprogramm aus erzeugt werden kann.
Das Beispielprogramm dient lediglich zur Demonstration der Prozeduren. Es besitzt keinen praktischen Hintergrund.

Übergroße Arrays

In bestimmten Fällen sind für ein Programm extrem große Arrays erforderlich, so beispielsweise zur Lösung von Gleichungssystemen mit vielen Unbekannten (Analyse elektrischer Netzwerke, Optimierungsprobleme) oder beim Abspeichern eines Leiterplattenentwurfes in Auto-Routing-Programmen. In derartigen Fällen kann das im Programm heap3 (siehe S. 267) angedeutete Verfahren genutzt werden.
Das benötigte Array wird in einzelne Zeilen aufgelöst und diese dynamisch vereinbart. Die Adresse jeder so vereinbarten Zeile wird in einem weiteren, ebenfalls dynamisch vereinbarten Array abgespeichert.
Für den Zugriff auf die einzelnen Elemente des Arrays stehen zwei Prozeduren zur Verfügung, die dem Anwender die Manipulation mit Zeigern ersparen.
Bei der Arbeit mit dem Programm ist zu beachten, daß beim 16-Bit-Turbo-Pascal die Standardfunktion memavail die Größe des zur Verfügung stehenden Speichers in Einheiten zu 16 Byte (sogenannte Paragraphen) zurückgibt. Alle Berechnungen, in denen diese Funktion genutzt wird, müssen dementsprechend angepaßt werden. Dabei ist die Gefahr der Überschreitung des Wertebereiches von Integer-Variablen zu berücksichtigen.
Auf 16-Bit Systemen werden zur Abspeicherung eines Zeigers 4 Byte benötigt (Startadresse des 64-kByte-Segments und Offset innerhalb des Segmentes). Dieser Platz ist bei der Planung der Heap-Nutzung zu berücksichtigen, da er zumindest bei den beiden ersten Programmen nicht zu vernachlässigen ist.

```
  procedure disksuch(var eintrag:et);
  {anwenderspezifische prozedur zum suchen eines datensatzes auf disk-datei}
  {entsprechend eines merkmales aus "eintrag"}
  {wenn satz gefunden, dann vervollstaendigen von "eintrag", ansonsten fuellen}
  {mit fehlstring zur fehlerkennzeichnung}
  begin
  end;

begin  {procedure suchen}
  sucheintrag:=erster_eintrag;
  while ((sucheintrag<>nil) and (sucheintrag^.matnr<>material.matnr)) do
    sucheintrag:=sucheintrag^.next;
  if sucheintrag^.matnr=material.matnr then             {wenn nummer gefunden}
  begin
    material:=sucheintrag^;                        {eintrag vollstaendig fuellen}
    nach_vorne(sucheintrag);
  end
  else
  begin
    disksuch(material);
    if material.matnr <> fehlstring then einsortieren(material);
  end;
end;

begin {Hauptprogramm}
  erster_eintrag:=nil;
  .
  .
  suchen(eintrag);
  .
  .
end.


program heap2;
const maxarray = 10000;              {maximale anzahl von datensaetzen im heap}
      freemem  =  1000;          {paragraph's (16 Byte) die frei bleiben sollen}

type z = ^struktur;                                 {zeiger auf einen datensatz}
     m = ^a;                                     {zeiger auf array mit zeigern}
     a = array[0..maxarray] of z;             {array mit zeigern auf datensaetze}
     s = string[255];
     struktur = record                              {typ-definition fuer datensatz}
                  materialnummer:string[10];                 {anwenderspezifisch}
                  preis:real;
                  summe:real;
                  anzahl:integer;
                end;

var datei:file of struktur;
    datensatz:struktur;
    alles_im_heap:boolean;                         {zeigt an, ob datei im heap ist}
    dat_eof:boolean;                                 {eof-kennzeichen fuer datei}
    geaendert:boolean;                            {zeigt schreibvorgang in heap an}
    adresse:m;                                {array mit zeigern auf datensaetze}
    arraygroesse:integer;         {augenblickliche anzahl von datensaetzen im heap}
    satzzeiger:integer;   {nummer des aktuellen datensatzes, aehnlich filepos()}

procedure fehler(text:s);                    {fehlerausschrift, programmabbruch}
begin
  writeln;writeln;
  writeln(text);
  writeln;writeln;
  halt;
end;

procedure dat_open;        {eroeffnen datei, in heap einlesen, wenn platz reicht}
var zeiger:z;
    size:real;
begin
  reset(datei);
  size:=filesize(datei);     {real-zahl nutzen, bei integer ueberlauf moeglich}
  if (size * sizeof(datensatz)/16 + 0.25 * maxarray < memavail-freemem) and
     (filesize(datei) <= maxarray) then
  begin                                                     {datei in heap einlesen}
    new(adresse);                                       {array fuer zeiger erzeugen}
    while not eof(datei) do
    begin
      read(datei,datensatz);                                     {einen satz lesen}
      new(zeiger);                                  {platz fuer satz im heap schaffen}
      adresse^[filepos(datei)-1]:=zeiger;              {adresse im array abspeichern}
      adresse^[filepos(datei)-1]^:=datensatz;                 {dyn. variable fuellen}
    end;
    alles_im_heap:=true;
  end
  else
    alles_im_heap:=false;
  arraygroesse:=filesize(datei);                          {aktuelle dateigroesse }
  satzzeiger:=0;
  dat_eof:=false;
  geaendert:=false;
end;


procedure dat_close;     {schlieszen der diskettendatei bzw. auslesen des heap}
var n:integer;
begin
  if alles_im_heap and geaendert then
  begin
    rewrite(datei);
     for n:=0 to arraygroesse-1 do
      write(datei,adresse^[n]^);
  end;
  close(datei);
end;
```

```
procedure dat_seek(n:integer);                          {setzen des satzzeigers}
begin
  if alles_im_heap then
  begin
    if arraygroesse < maxarray then
    begin
      if ((n >=0) and (n<=arraygroesse))
         then satzzeiger:=n
         else fehler('Ausserhalb Dateigrenzen');
      if ((satzzeiger = arraygroesse) and
         ((memavail-freemem) < sizeof(datensatz)/16+1)) then
            fehler('Heap zu klein');
    end
    else fehler('Satzzahl zu gross');
  end
  else
  begin
    if ((n>=0) and (n<=filesize(datei))) then
    begin
      seek(datei,n);
      satzzeiger:=n;
    end
    else fehler('Ausserhalb Dateigroesse');
  end;
end;

procedure dat_read(var datensatz:struktur);
{einlesen des mit satzzeiger benannten datensatzes in uebergebene variable}
begin
  if alles_im_heap then
  begin
    if satzzeiger=arraygroesse then fehler('Lesen nach File-Ende');
    datensatz:=adresse^[satzzeiger]^;
    satzzeiger:=satzzeiger+1;
    dat_eof:= satzzeiger=arraygroesse;
  end
  else
  begin
    read(datei,datensatz);
    satzzeiger:=filepos(datei);
    dat_eof:=eof(datei);
  end;
end;
```

```
procedure dat_write(var datensatz:struktur);

var zeiger:z;
begin
  if alles_im_heap then
  begin
    if satzzeiger=arraygroesse then
    begin
      new(zeiger);
      adresse^[satzzeiger]:=zeiger;
      arraygroesse:=satzzeiger+1;
    end;
    adresse^[satzzeiger]^:=datensatz;
    dat_seek(satzzeiger+1);
    dat_eof:=satzzeiger=arraygroesse;
    geaendert:=true;
  end
  else
  begin
    write(datei,datensatz);
    satzzeiger:=filepos(datei);
    dat_eof:=eof(datei);
  end;
end;

begin
  assign(datei, probe.dat');
  dat_open;
  while not dat_eof do
  begin
    dat_read(datensatz);
    with datensatz do
    begin
      summe:=preis*anzahl;
    end;
    dat_seek(satzzeiger-1);
    dat_write(datensatz);
  end;
  dat_close;
end.
```

```
program heap3;
const xmax=100;                                         {groesse des arrays}
      ymax=100;
      freemem = 1000;            {platz, der im heap frei gelassen werden soll}

type feldzeiger = ^adressen;           {zeiger auf array mit zeigern auf zeilen}
     z          = ^zeile;                     {zeiger auf eine zeile des arrays}
     feldtyp    = real;                     {typ-definition fuer feldelemente}
     adressen   = array[1..ymax] of z;         {array mit zeigern auf zeilen}
     zeile      = array[1..xmax] of feldtyp;                {zeile des arrays}

var  zeiger:z;                                        {weist auf eine zeile}
     feld:feldzeiger;                   {weist auf array mit zeigern auf zeilen}
     a,b:integer;
     ok:boolean;                        {zeigt ordnungsgamaesze Ausfuehrung an}
     feldelement:feldtyp;

function wert(x,y:integer):feldtyp;                 {holen des elementes x,y}
begin
  ok:=true;
  if ((x in [1..xmax]) and (y in [1..ymax]))
    then wert:=feld^[y]^[x]
    else ok:=false;
end;

procedure schreib(x,y:integer;b:feldtyp);     {belegen des elementes x,y mit b}
begin
  ok:=true;
  if ((x in [1..xmax]) and (y in [1..ymax]))
    then feld^[y]^[x]:=b
    else ok:=false;
end;

procedure initial;                                {anlegen des arrays im heap}
var n:integer;
    groesse:real;
begin
  new(feld);                             {array mit zeigern auf zeilen anlegen}
  groesse:=xmax;             {wandeln integer in real, da sonst ueberlauf}
  groesse:=(groesse * ymax * sizeof(feldelement)+sizeof(adressen))/16;
  if groesse < memavail - freemem then             {test, ob platz reicht}
  begin
    for n:=1 to ymax do
    begin
      new(zeiger);                                       {neue zeile anlegen}
      feld^[n]:=zeiger;                      {adresse der neuen zeile merken}
    end;
    ok:=true;
  end
  else
  begin
    writeln('Zu wenig Speicherplatz');
    ok:=false;
  end;
end;
```

```
begin {heap3}
  initial;
  if not ok then halt;
  for a:=ymax-3 to ymax do
    for b:=xmax-3 to xmax do
      schreib(a,b,a+b);
  for a:=ymax-3 to ymax do
    for b:=xmax-3 to xmax do
    begin
      feldelement:=wert(a,b);
      writeln(feldelement);
    end;
end.
```

Kleines Lexikon der Mikrorechentechnik

M
wie Maskenprogrammierung



Zeichnung: Dahmen

Typenraddrucker mit IFSS an V.24-Schnittstelle

Manfred Kramer
Bauakademie der DDR, Institut für Heizung, Lüftung und Grundlagen der Bautechnik

Die Typenraddrucker der Reihe SD 1152 vom VEB Büromaschinenwerk Sömmerda weisen gegenüber den am PC 1715 zumeist eingesetzten Matrixdruckern Unterschiede auf, die in einigen Anwendungsfällen von Interesse sind. Neben der besseren Qualität des Druckbildes, die für Korrespondenz und Textverarbeitung wichtig ist, interessiert bei Anwendern in der Verwaltung auch die Möglichkeit, mit mehreren Durchschlägen und auf 2 unabhängigen Formularen drucken zu können.

Es entstand daher der Wunsch, auch diese Drucker wahlweise am PC 1715 zu betreiben. Vom Hersteller wird hierfür zwar eine nachrüstbare Baugruppe angeboten, jedoch läßt sich das Problem auch mit wesentlich geringerem Aufwand unter Nutzung der am Gerät vorhandenen, aber meist ungenutzten V.24-Schnittstelle lösen.

Bild 1 zeigt die Schaltung eines Pegelwandlers, mit dem die Spannungen der V.24-Schnittstelle an die 20-mA-Stromschleife angepaßt werden. Da die V.24 mit Pegeln von 3 V und 12 V arbeitet, reichen am Eingang die Spannungen von +5 V und −5 V für eine „vorschriftsmäßige" Ansteuerung aus. Sie werden am Stecker für die optionalen Floppy-Laufwerke entnommen. (Vorsicht: Druckfehler in Dokumentation!)

Die Schaltung ist auf einer kleinen Leiterplatte in einem Gehäuse aus kupferkaschiertem Leiterplattenmaterial von der Größe einer Streichholzschachtel untergebracht und steckt unmittelbar hinten am PC.

Die Programme der beiden Schnittstellen unterscheiden sich etwas: Während bei der V.24 die Abfrage der Quittungsleitungen die Zeichenübertragung in Richtung Drucker steuert, erfolgt bei der Stromschleife ein Datenverkehr in beiden Richtungen, so daß sowohl Ausgabe- als auch Eingabekanal der Schnittstelle nötig sind. Deshalb ist die eigentliche Druckerschnittstelle für die Stromschleife auch ungeeignet, da sie nur einen Ausgabekanal aufweist.

Bild 2 zeigt das Programm.

Auf eine Installation im Betriebssystem wurde der Einfachheit halber verzichtet, es wird bei Bedarf auf eine nicht benutzte Stelle im Speicher geladen und seine Startadresse in der Sprungtabelle des BIOS anstelle der installierten Druckroutine eingetragen.

Durch ein ähnliches, noch einfacheres Programm kann man die Adresse der installierten Druckroutine ohne Kaltstart wieder eintragen und damit zwischen beiden Druckern per Software umschalten.

Bild 2 Listing ▸

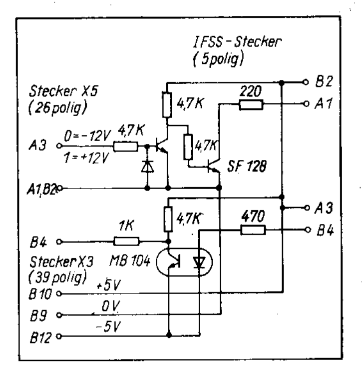


Bild 1 V.24-IFSS-Anpassung für PC 1715

```
ASEG

SCPX    EQU     1
IF      SCPX
BIOS    EQU     0D900H
ZIEL    EQU     0EF7EH
ELSE                            ;CPA
BIOS    EQU     0D400H
ZIEL    EQU     0F770H
ENDIF
SIOBS   EQU     0FH
SIOBD   EQU     0DH
CTC     EQU     9H

ORG     100H
;übertragen des Druckprogramms in das BIOS
        LD      HL,AN
        LD      DE,ZIEL
        LD      BC,LLO                  ;Länge
        LDIR
;Druckprogramm in Sprungtabelle eintragen
        LD      HL,LO
        LD      (BIOS+010H),HL
;listerstatus f. ws eintragen
        ld      hl,bios+2dh
        ld      a,3eh
        ld      (hl),a
        inc     hl
        ld      a,0ffh
        ld      (HL),a
        inc     hl
        ld      a,0c9h
        ld      (hl),a
;Initialisierung CTC
INIT:   LD      A,5
        OUT     (CTC),A
        LD      A,1
        OUT     (CTC),A
;Initialisierung SIO
        LD      B,7
        LD      C,SIOBS                 ;Statusport
        LD      HL,SDILT                ;Tabelle
        OTIR
        LD      C,7FH
        CALL    LO
        XOR     A
        LD      C,A
        JP      0                       ;Warmstart

SDILT:  DEFB    18H,4,45H,5,28H,3,41H
                ;Cx16,1 Stopp, 7 Bit, ungerade Parität

AN      EQU     $
LO:     PUSH    BC
        IN      A,(SIOBS)
        BIT     0,A
        JR      Z,SDR3
SDR1:   IN      A,(SIOBD)
        CP      91H     ;Freigabe Drucker
        JR      Z,SDR3
        CP      94H     ;Status Drucker
        JR      Z,SDR4
SDR2:   IN      A,(SIOBS)
        BIT     0,A
        JR      Z,SDR2
        JR      SDR1
SDR3:   IN      A,(SIOBS)
        BIT     2,A     ;Sendepuffer leer ?
        JR      Z,SDR3
;Zeichenausgabe
        LD      A,C
        OUT     (SIOBD),A
SDR4:   POP     BC
        RET
LLO     EQU     $-LO
END
```

⊠ KONTAKT ☏

Bauakademie der DDR, Institut für Heizung, Lüftung und Grundlagen der Bautechnik, Plauener Straße 163, Berlin, 1092; Tel.: 37 83 22 46

●

Digitale Meßumformer in der Verfahrenstechnik

Hans O. Engel
Honeywell Regelsysteme GmbH, Produktionsbereich Feldgeräte

Fallende Preise für Mikroprozessoren in Verbindung mit einer ständig steigenden Leistung und Zuverlässigkeit haben in den letzten Jahren einen beispiellosen Innovationsschub auf allen Gebieten der Technik bewirkt. Während moderne Prozeßleitsysteme in der industriellen Verfahrenstechnik längst zum Stand der Technik geworden sind und eine immer größere Verbreitung finden, waren die Feldgeräte lange Zeit Stiefkinder der Automation.

Der folgende Beitrag versucht die Vorteile digitaler Feldgeräte deutlich zu machen, einen Vergleich mit der konventionellen Analogtechnik herzustellen und eine Unternehmensstrategie zu erläutern, die einen „Bruch", das heißt Inkompatibilität bei der Umstellung auf digitale Meßumformer vermeidet. Schließlich wird kurz auf die Problematik bei der Normung eines internationalen Feldbusses eingegangen.

Grenzen der Analogtechnik

Es hat mehr als 10 Jahre gedauert, bis der zunächst in den USA gebräuchliche 4-20-mA-Standard sich weltweit durchsetzte und schließlich im Jahre 1975 von der IEC (International Electrical Commission) genormt werden konnte. Diese Maßnahme gilt auch heute noch als ein Durchbruch zur Vereinheitlichung der unterschiedlichen Signalbereiche, wurden doch auf einmal Geräte verschiedener Hersteller kompatibel, und der Anwender konnte seine Regelkreise nun unbeschadet mit Geräten verschiedener Hersteller bestücken.

Die Analoggeräte sind seit dieser Zeit ständig verbessert worden, und viele Anwender sehen auch heute noch überhaupt keinen Grund, zur Digitaltechnik überzuwechseln. Laufen denn seine Anlagen nicht zufriedenstellend? Ist nicht erst im vergangenen Jahr ein Rekordüberschuß erwirtschaftet worden? Warum also ein Risiko eingehen, wenn keine zwingenden Gründe vorliegen und alles in Ordnung zu sein scheint. Dieser als *Alles-OK-Haltung* bekannte Standpunkt trübt leider manchmal den Blick für die Wirklichkeit. Die Unzulänglichkeit analoger Meßumformer wurde erst in jüngster Zeit ins Bewußtsein der Meß- und Regeltechnik gerufen, seit es digitale Geräte mit verbesserter Funktionalität und zum Teil völlig neuen Geräteeigenschaften gibt. So werden die Grenzen der Analogtechnik insbesondere bei folgenden Bewertungskriterien sichtbar:

- Die Alterung der Bauelemente bewirkt eine *Verschiebung von Bereich und Nullpunkt*.
- Die relativ hohe Leistungsaufnahme und der Zwang zur Kompaktheit *reduziert die Lebensdauer* der Geräte (Badewannenkurve).
- Der *Informationsgehalt* des Analogsignals ist nur gering. Das Zweileitersystem dient ausschließlich der Übertragung der Prozeßvariablen. Weitere Informationen sind nicht abrufbar.
- Alle Meßumformer weisen *systematische Fehler* auf, die zwar durch konstruktive Maßnahmen minimiert, aber nie völlig ausgeschaltet werden können.

● Die genaue, *driftarme Verstärkung und Umwandlung* des Sensorsignals stellt ein weiteres Problem dar.
● Meßungenauigkeiten erhöhen sich bei einer Bereichsunterteilung in der Regel proportional. Dies erfordert eine *Begrenzung der Teilung* auf ein Verhältnis von etwa 1 : 6.
● Selbst bei sehr genauen Analoggeräten bleibt das Problem der Auflösung bestehen, weil auf der Empfängerseite der *Auflösung des 4-20-mA-Signals* Grenzen gesetzt sind.

Statistische Untersuchungen an Differenzdruck-Meßumformern, die etwa auf die Hälfte des maximalen Meßbereiches eingestellt wurden und bei Änderungen des statischen Drucks von etwa 20 bar sowie veränderlichen Umgebungstemperaturen zwischen 0 und 80°C arbeiteten, ergaben folgende Meßungenauigkeiten:
– bei Referenzbedingungen (20 °C, statischer Druck konstant) ≈0,5 %
– bei veränderlichen Umweltbedingungen 1,7 %
– bei Umweltbedingungen und Langzeitdrift (1 Jahr) 4,5 %

Digitale Meßumformer

Die Einsatzmöglichkeiten von Mikroprozessoren wurden im Laufe der Zeit ständig erweitert und finden seit 1983 auch erstmals bei Meßumformern (MU) praktische Anwendung /1/. Zunächst müssen aber die Signale der analogen Meßwertaufnehmer oder Sensoren in eine für den Mikroprozessor verständliche Form gebracht, das heißt digitalisiert werden. Die prinzipiellen Unterschiede

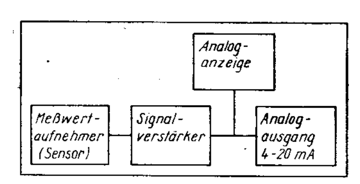


Bild 1 Prinzip einer analogen Meßwertverarbeitung

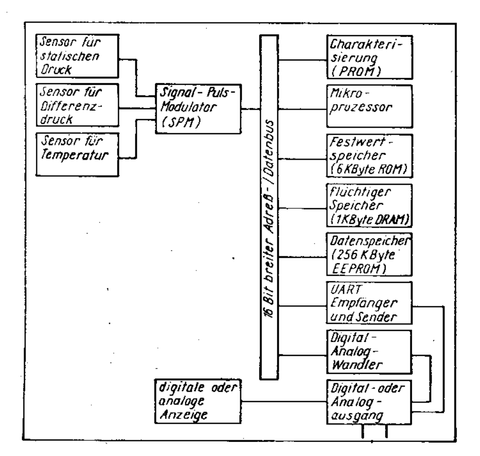


Bild 2 Prinzip eines digitalen Meßumformers ST 3000 mit (wahlweise) digitalem oder analogem Ausgang

zwischen analogen und digitalen Meßumformern sind in den Bildern 1 und 2 dargestellt.
Bild 2 zeigt das Schema einer realisierten digitalen Signalverarbeitung am Beispiel eines MUs für Differenzdruck. Der dreifach integrierte Analogsensor liefert seine Signale an einen Signal-Puls-Modulator (SPM), der die Aufgaben eines Signalverstärkers und eines Multiplexers zusammenfaßt. Ein interner 16-Bit-Datenbus verbindet den Signal-Puls-Modulator mit dem Mikroprozessor und den anderen Digitalbausteinen.
Die unvermeidlichen und systematischen Meßungenauigkeiten eines MUs für Differenzdruck werden für jedes Gerät individuell erfaßt und als *Charakterisierungsdaten* in einem Festwertspeicher abgelegt. Die Haupteinflußgrößen Temperatur und statischer Druck werden gemessen, so daß der Mikroprozessor das Ausgangssignal entsprechend der abgelegten Fehlerkurve des MUs korrigieren kann. Dadurch wird eine bisher unerreicht hohe Meßgenauigkeit ermöglicht /2/.
Die Vorteile digitaler Meßumformer lassen sich am besten anhand der Grenzen analoger Geräte verdeutlichen:
① Bereich und Nullpunkt sind weitaus stabiler, das heißt, die Wiederholgenauigkeit – die wichtigste Eigenschaft aller MUs – wird entscheidend verbessert, weil die Abhängigkeit von unvollkommenen Bauelementen (Alterung) entfällt.
② Die Meßungenauigkeiten lassen sich relativ einfach kompensieren. Werte von maximal 0,1 % sind heute ohne weiteres erreichbar. Das verlangt allerdings eine recht aufwendige, individuelle Erfassung der Fehlerkurven und die Ablage der Korrekturfaktoren in einem PROM.
③ Die Zuverlässigkeit hochintegrierter Schaltkreise in Verbindung mit geringer Leistungsaufnahme (CMOS-Technik) erreicht heute Werte, die vor einiger Zeit noch für unmöglich gehalten wurden. Der am häufigsten angewendete Maßstab *Mean Time between Failures (MTBF)* ergibt beispielsweise für den Differenzdruck-Meßumformer ST 3000 (Baujahr 1988) einen Wert von über 400 Jahren /3/!
④ Der Informationsgehalt ist trotz Beibehaltung des bewährten Zweileitersystems hoch. Es kann bei digitalen Geräten nicht nur die Hauptmeßgröße (das heißt der Istwert) übertragen, sondern eine Vielzahl weiterer Informationen (zum Beispiel der Status des MUs) abgefragt werden.
⑤ Die digitale Architektur der MUs ermöglicht eine Kommunikation über große Entfernungen. Die wichtigsten Prüfungen (Diagnose) sind von der Warte aus möglich. Das Aus- und Wiedereinbauen der MUs entfällt bei der Wartung. Abgesehen von erheblichen zeitlichen Einsparungen reduziert sich das Risiko für das Wartungspersonal, wenn beispielsweise die Meßumformer schwer zugänglich sind. Problematisch ist auch die Berührung mit gefährlichen oder giftigen Prozeßmedien, denen das Personal bei einer herkömmlichen Wartung und Instandhaltung ausgesetzt sein kann.
⑥ Die Konfiguration der MUs ist sehr einfach zu verändern, wenn die Betriebsbedingungen dies erfordern. Auch hier erweist sich die Kommunikationsmöglichkeit mit den MUs als sehr nützlich. Noch wichtiger ist allerdings die Tatsache, daß die Meßbereiche digitaler MUs mit analogem Ausgangssignal in sehr weiten Grenzen veränderbar sind. Erlauben analoge Geräte – mit Rücksicht auf vertretbare Meßungenauigkeiten – eine Bereichsteilung von etwa 1 : 6, so sind bei digitalen MUs mit analogem Ausgangssignal Spreizungen bis 1 : 400 möglich. Dadurch wird die Anzahl der erforderlichen Typen erheblich eingeschränkt, was sich günstig auf die Lagerhaltung auswirkt.
⑦ Meßumformer mit digitalem Ausgangssignal können auf eine Bereichseinstellung gänzlich verzichten. Je nach Auflösegenauigkeit des A/D-Wandlers sind selbst bei Meßbereichen, die nur ein Tausendstel des MU-Meßbereiches betragen, noch befriedigende Resultate zu erzielen, wie die folgende Tafel zeigt:

Auflösung des A/D-Wandlers in Bit	Anzahl der Stufen	Auflösung in %
6	64	1,56
8	256	0,39
10	1024	0,098
12	4096	0,024
14	16384	0,0061
16	65536	0,0015

Internationale Norm für einen Feldbus

Seit mehreren Jahren gibt es Bemühungen, die Randbedingungen zur Nutzung eines internationalen Standards festzuschreiben. Leider ist die Thematik so komplex und vielschichtig, daß eine Einigung bisher nicht gelungen ist. Allerdings ist zu erwarten, daß zumindest ein Pflichtenheft für eine vorläufige Norm bis 1990 fertiggestellt wird. Abgesehen von „privaten" Feldbussen, gibt es eine Reihe von Kandidaten mit internationaler Bedeutung, die einen Normentwurf vorgestellt und bereits praktische Erfahrungen gesammelt haben. Ohne Anspruch auf Vollständigkeit zu erheben, gelten folgende Systeme als die wichtigsten Feldbus-Kandidaten:
● Profibus (BRD: Vornorm DIN 19245)
● FIP-Bus (Frankreich)
● EUREKA-Bus (Gemeinschaftsprojekt: BRD, Frankreich, Großbritannien, Italien, Norwegen, Finnland)
● Mil-Bus (Großbritannien: Mil.-Std. 1553B)
● SP-50 Bus (USA)
Aufgabe der IEC wird es sein, eine internationale Norm zu erarbeiten, die den Minimalforderungen der Industrie – die in den Spezifikationen der verschiedenen Kandidaten zum Ausdruck kommt – gerecht wird (IEC/SC 65B/WG 6).
In diesem Zusammenhang wird auch die Kompatibilität mit *MAP (Manufacturing Automation Protocol)* diskutiert, das von General Motors zum Zweck einer herstellerneutralen „offenen Kommunikation" initiiert wurde. Es basiert auf dem bekannten ISO/OSI-Siebenschichtenmodell, das auch die Grundlage für den internationalen Feldbus bildet (siehe auch MP 1/1989, Seite 7).
Die allgemeinen Anforderungen an einen Feldbus für die Prozeß- und Verfahrensindustrie werden nachfolgend grob beschrieben /4/:
– bitserielle Datenübertragung (bidirektional)
– vorzugsweise verdrillte (geschirmte) Zweidrahtleitung
– einfache, rückwirkungsfreie Ankopplung
– galvanische Trennung der einzelnen Bus-Teilnehmer
– Hilfsenergieversorgung über Bus-Kabel
– Ex-Schutz durch „eigensichere" Stromkreise
– als Option Maßnahmen zur Erhöhung der Verfügbarkeit
– einheitliche (mechanische) Anschlußtechnik
– elektromagnetische Verträglichkeit (EMV)
– möglichst hohe Übertragungsraten, kurze Protokolle
– Daten-Transport, -Sicherung, -Protokoll nach IEC-Norm
– niedrige Anschlußkosten.

Eine voraussichtliche Feldbusstruktur wird maximal 32 Teilnehmer bei Entfernungen bis zirka 1,5 km beinhalten. Wahrscheinlich wird einer zentralen Bussteuerung gegenüber einer dezentralen (token ring) der Vorzug eingeräumt. Es wird keinen Universalbus geben, sondern zwei verschiedene Systeme für unterschiedliche Anwendungen:

- **Bus H1:** für normale Anwendungen mit mittleren Übertragungsraten; langfristiger Ersatz für 4-20-mA-Analogsignal
- **Bus H2:** Bus für schnelle, komplexe Anwendungen in der Fabrikautomation.

Eine Kompatibilität mit einem bereits bestehenden (privaten) System ist unwahrscheinlich. Deshalb kommt der Bereitschaft des Herstellers, seine Geräte bei der Verwirklichung eines einheitlichen Standards entsprechend zu modifizieren und das Investment des Anwenders zu erhalten, große Bedeutung zu.

Literatur

/1/ Bradshaw, A. T.: ST 3000 – „Intelligente" Druckmeßumformer. rtp (1983) 12, S. 531

/2/ Borst, W.; Pfündlin, E.; Ziesemer, M.: „Intelligente" Meßumformer für die Verfahrenstechnik. atp (1987) 11, S. 526

/3/ Honeywell Bulletin CW 8901001, 10. 1. 1989

/4/ Pfleger, J.: Kommunikationssystem Feldbus, atp (1986) 5, S. 223

MS-DOS-Tip

Kursormanipulation

Eberhard Schmidt
WTZ der Land- und Nahrungsgüterwirtschaft Frankfurt (Oder)

Programmiersysteme wie Turbo-Pascal oder Turbo-C enthalten Bibliotheken, die den direkten Zugriff auf die Ressourcen des Betriebssystems MS-DOS unterstützen. Die hierin vordefinierten Interrupt-Routinen ermöglichen neben dem Aufruf der MS-DOS-Funktionen über den Software-Interrupt 21H auch die Ansprache des ROM-BIOS (Interrupts 10H bis 1FH).

Über den Interrupt 10H sind dem Programmierer zum Beispiel die Bildschirm-Service-Routinen des Betriebssystems zugänglich. So könnte es eventuell sinnvoll erscheinen, in einzelnen Programmpassagen den Kursor zu verstecken oder ihn als Block zu betonen. Der Kursor-Modus wird über die Subfunktion 1 des Bildschirm-Services gesetzt. Dabei sind im Register CX die Start- und die End-Linie des Kursors zu definieren. Hierin liegt nun eine gewisse Tücke in bezug auf die Kompatibilität von Programmen, die diese Funktion nutzen, denn die End-Linie des Standard-Kursors *Unterstrich* hat beim grafikfähigen Bildschirm des A 7150 (oder Schneider-PC) die Nummer 7, während sie beim alphanumerischen Bildschirm des EC 1834 die Nummer 12 hat.

Es ist deshalb angebracht, den Standard-Modus des Kursors vor dessen erster Änderung zu sichern. Dies könnte durch Abfrage der BIOS-Konstanten CURSOR_MODE geschehen, die im MS-DOS 3.2 und 3.3 auf der Adresse 40 : 60H steht. Eine zweite, hier demonstrierte Möglichkeit ist der Einsatz der Subfunktion 3. Die beigefügten Beispiele für Turbo Pascal ab Version 4.0 und C demonstrieren diese Verfahrensweise unter Nutzung der Eigenschaften der statischen Variablen First_Call und Cursor_Size. Die Pascal-Prozedur Cursor_Modus (Bild 1) benutzt einen Datentyp Registers und die Prozedur INTR. Beide sind in der UNIT DOS definiert, die demzufolge in der ersten Zeile des Beispielprogramms dem Compiler genannt werden muß.

Analog dazu enthält die C-Bibliothek dos.h den Typ REGS und die Funktion int86 (Bild 2).

Die Subfunktion 3 wird nur beim ersten Aufruf von Cursor_Modus aktiviert, um die geräteabhängigen Parameter für die Cursor-Modi *Unterstrich* und *Block* zu setzen. Dabei wird vorausgesetzt, daß der Standard-Modus *Unterstrich* eingestellt ist. Er sollte deshalb stets beim Verlassen des Anwenderprogramms wieder zurückgesetzt werden.

✉ KONTAKT ☏

WTZ der Land- und Nahrungsgüterwirtschaft, Lebuser Chaussee 11, Frankfurt (Oder), 1200; Tel. 31 12 18


```
USES DOS ;
CONST Unterstrich = 0 ;
      Block       = 1 ;
      unsichtbar  = 2 ;

(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)
PROCEDURE Cursor_Modus ( C_Typ : BYTE ) ;
         {============}
CONST  Hidden = $2020 ;
       First_Call   : BOOLEAN = TRUE ;
       Cursor_Size  : ARRAY [ Unterstrich .. unsichtbar ]
                                     OF WORD = ( 0,0,0 ) ;
       Bildschirm_Service = $10 ;
VAR Regs    : Registers ;
BEGIN
   IF First_Call THEN
    BEGIN
     regs.ah := 3 ; regs.bx := 0 ;
     INTR ( Bildschirm_Service , regs ) ;
     Cursor_Size [ Unterstrich ] := Regs.cx ;
     Cursor_Size [ Block       ] := Regs.cl ;
     Cursor_Size [ unsichtbar  ] := Hidden   ;
     First_Call := FALSE
    END ;
   regs.ah := 1 ;  regs.bx := 0 ;
   regs.cx := Cursor_Size [ C_Typ ] ;
   INTR ( Bildschirm_Service , regs )
END ;
(* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *)

PROCEDURE warten ;
VAR st : STRING [ 255 ] ; BEGIN READLN ( st ) END ;

BEGIN
   WRITE ( 'Standard-Cursor     : ' ) ; warten ;
   Cursor_Modus ( unsichtbar  ) ;
   WRITE ( 'Cursor unsichtbar   : ' ) ; warten ;
   Cursor_Modus ( Block       ) ;
   WRITE ( 'Cursor Block        : ' ) ; warten ;
   Cursor_Modus ( Unterstrich ) ;
   WRITE ( 'Cursor Unterstrich  : ' ) ; warten
END.
```

***Bild 1** Listing des Turbo-Pascal-Beispielprogramms*

```
#include <dos.h>
#define Unterstrich 0
#define Block       1
#define unsichtbar  2

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
  Cursor_Modus ( short C_Typ )
/*===========*/
#define TRUE  1
#define FALSE 0
#define Hidden 0x2020
#define Bildschirm_Service 0x10
{
  union REGS regs ;
  static short First_Call = TRUE ;
  static unsigned int   Cursor_Size[] = { 0,0,0 };
  if ( First_Call )
     {
       regs.h.ah = 3 ;    regs.x.bx = 0 ;
       int86 ( Bildschirm_Service , &regs , &regs ) ;
       Cursor_Size [ Unterstrich ] = regs.x.cx ;
       Cursor_Size [ Block       ] = regs.h.cl ;
       Cursor_Size [ unsichtbar  ] = Hidden ;
       First_Call = FALSE ;
     }
  regs.h.ah = 1 ;    regs.x.bx = 0 ;
  regs.x.cx = Cursor_Size [ C_Typ ] ;
  int86 ( Bildschirm_Service , &regs , &regs ) ;
}
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

warten()
{  char st [ 255 ] ;    scanf ( "%s" , st ) ;  }

main()
{
    printf ( "Standard-Cursor    : " ) ; warten() ;
    Cursor_Modus ( unsichtbar  ) ;
    printf ( "Cursor unsichtbar  : " ) ; warten() ;
    Cursor_Modus ( Block       ) ;
    printf ( "Cursor Block       : " ) ; warten() ;
    Cursor_Modus ( Unterstrich ) ;
    printf ( "Cursor Unterstrich : " ) ; warten() ;
}
```

***Bild 2** Listing des Turbo-C-Beispielprogramms*

Einführung in Forth-83

Teil 4

Dr. Hartmut Pfüller (Leiter),
Dr. Wolfgang Drewelow, Dr. Bernhard Lampe,
Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

4. Die virtuelle Forthmaschine

Diese Folge bietet in aller Kürze eine Darstellung der inneren Funktion eines Forthsystems. Während der Durchschnittsprogrammierer diese Informationen im allgemeinen entbehren kann, wird dem interessierten Leser gezeigt, wie die Leistungsfähigkeit von Forth auf wenigen, überraschend einfachen Konzepten beruht. In den vorangegangenen Folgen stand die Beschreibung derjenigen Komponenten von Forth im Mittelpunkt, die die für den Programmierer sichtbare *Oberfläche* des Systems bilden. Bevor in den weiteren Folgen Möglichkeiten zur *Erweiterung* des Systems vorgestellt werden, versucht dieser Beitrag, die *interne Arbeitsweise* eines Forthsystems verständlich zu machen. Im ersten Abschnitt wird der Aufbau des Wörterbuchs behandelt. Die Erläuterung der internen Struktur von Worten macht die Methode transparent, nach der Programme auch über indirekte Bezugnahmen aufgerufen werden können. Im zweiten Abschnitt wird die interne Arbeitsweise von Forth vorgestellt. Auf dieser Grundlage wird dann als drittes die häufig notwendige Einbindung von Maschinencode eingeführt.

4.1. Die interne Wortstruktur

4.1.1. Komponenten eines Wortes

Die fundamentalen Bausteine eines Forthsystems sind die Worte. Sie müssen unter zwei Aspekten betrachtet werden. Einerseits sind Worte ausführbare Softwaremoduln, wobei durch ihre Struktur verborgen wird, ob es sich um Datenmoduln (z. B. Variablen oder Konstanten) oder Programmoduln (Doppelpunktdefinition) handelt. Andererseits stellen die Worte auch über ihre Namen die Verbindung zum Bediener her. Entsprechend dieser funktionellen Zweiteilung besteht jeder Eintrag im Wörterbuch aus zwei gekoppelten Teilen, die *Kopf* und *Rumpf* genannt werden. Der Kopf enthält dabei alle Informationen, die für die Dialogarbeit erforderlich sind. Dazu gehören der Name des Wortes, der im dafür vorgesehenen *Namenfeld* abgespeichert wird, sowie die Einordnung des Wortes in die Suchreihenfolge, die durch das sogenannte *Verkettungsfeld* geschieht. Der Rumpf nimmt dagegen alle diejenigen Informationen auf, die für die Abarbeitung des Wortes erforderlich sind. Im einzelnen sind dies im *Parameterfeld* liegende *Parameter* des Wortes und – ganz wichtig (!) – der im sogenannten *Codefeld* gespeicherte Verweis auf die bei Aufruf des Wortes auszuführende Aktion (Bild 4.1). Durch den Standard Forth-83 wird die konkrete Anordnung der einzelnen Felder nicht festgelegt; als Beispiel für die Erläuterungen hier dient das System comFORTH 2. Da die Größe und die Reihenfolge der einzelnen

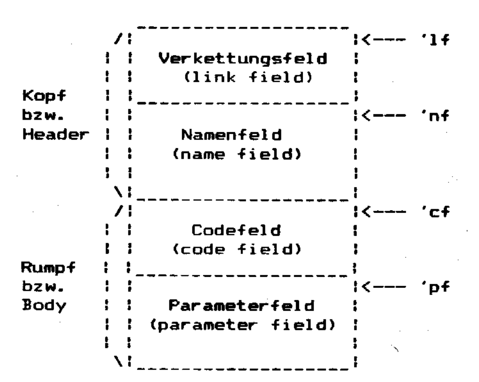


***Bild 4.1** Aufbau eines einzelnen Wörterbucheintrags; 'lf bedeutet Linkfeldadresse, 'nf Namenfeldadresse usw.*

Wortkomponenten in den Systemen verschiedener Hersteller oft unterschiedlich sind, bietet der Standard in einem experimentellen Vorschlag einen Wortschatz an, der wenigstens die Umrechnung der verschiedenen Anfangsadressen ineinander gestattet. Als einheitlicher Bezugspunkt wird dabei die Codefeldadresse 'cf benutzt. Aus ihr können mit Hilfe der Operationen **>LINK**, **>NAME** und **>BODY** die Link-, Namens- bzw. Parameterfeldadresse berechnet werden. Die entsprechenden Rückrechnungen erledigen die Worte **LINK>**, **NAME>** und **BODY>**. An sich stellen diese sechs Operationen bereits einen vollständigen Funktionensatz bereit, da auf dem Umweg über die Codefeldadresse aus jeder Adresse eine beliebige andere berechnet werden kann. Es ist jedoch so, daß aufgrund der Funktionsteilung zwischen Kopf und Rumpf beide Bestandteile nicht unbedingt in demselben Speicherbereich aufbewahrt werden müssen. So kann unter Umständen die physische Trennung von Kopf und Rumpf Vorteile bieten, zum Beispiel die Entlastung des 16-Bit-Adreßraums der Forthmaschine. Falls mit einer solchen Variante gearbeitet wird, ist für die Aufrechterhaltung der Systemfunktion nur ein Adreßverweis vom Kopf zum Rumpf zwingend erforderlich. Da in diesem Fall die beiden Funktionen >LINK und >NAME nicht mehr oder nur mit unverhältnismäßig hohem Aufwand implementiert werden können, ermöglichen die beiden Worte **N>LINK** und **L>NAME** die direkte Umrechnung von Namenfeld- und Verkettungsfeldadresse.

4.1.2. Suchoperationen

Suchoperationen über der Datenstruktur Wörterbuch sind an verschiedenen Stellen innerhalb des Systems erforderlich. Bei der Analyse von Quelltext muß z. B. der Textinterpreter untersuchen, ob die eingegebene Zeichenkette ein Wort ist, also ob ein entsprechender Eintrag in der aktuellen Suchordnung existiert. Für die Ausführung der Suchoperationen werden die im Kopf gespeicherten Informationen benutzt. Der Suchablauf im Wörterbuch wird zuerst durch die im Teil 2 des Kurses (siehe MP 5/1989) diskutierte Reihenfolge der zu durchsuchenden Vokabulare bestimmt. Innerhalb der Vokabulare legen die Verkettungsfelder die Suchreihenfolge fest. Die Suche erfolgt dabei umgekehrt zur zeitlichen Definitionsreihenfolge, das zuletzt definierte Wort wird also zuerst gefunden, danach das vorletzte usw. So wird auf einfache Weise erreicht, daß bei mehreren Definitionen gleichen Namens immer die jüngste als gültig gefunden wird. Erreicht wird dies durch eine Technik, die als verkettete Liste bezeichnet wird. Bild 4.2 zeigt den prinzipiellen Aufbau einer solchen Datenstruktur. Das Verkettungsfeld im Kopf eines Forthwortes enthält dementsprechend einen Adreßverweis zum Kopf des direkt zuvor in derselben Liste definierten Wortes. Im System comFORTH 2 ist an dieser Stelle die Linkfeldadresse des Vorgängers zu finden. Der Standard schreibt dies jedoch nicht vor, und es gibt auch Systeme, die jeweils auf die Namenfeldadresse des Vorgängers verweisen.

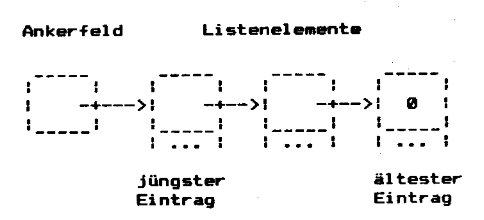


***Bild 4.2** Verkettete Liste*

Zum Vergleich der zu suchenden Zeichenkette mit dem Namen des Wortes dient das Namenfeld. Auch dessen Aufbau ist nicht standardisiert. Während comFORTH 2 den Namen in voller Länge abspeichert (siehe Bild 4.3), werden durch polyFORTH z. B. nur die ersten drei Zeichen abgespeichert, um Speicherplatz zu sparen und die Suchgeschwindigkeit zu erhöhen.

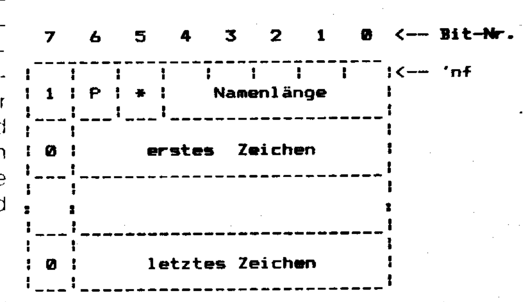


***Bild 4.3** Namenfeldaufbau im System comFORTH2*

Im Namenfeld sind zusätzlich zum Wortnamen einige weitere Informationen gespeichert. Dazu gehört insbesondere die Information, ob es sich um ein sogenanntes *bevorzugtes Wort* handelt, das unabhängig vom Systemzustand ausgeführt werden soll (immediate word). Die Diskussion dieser Worte ist Gegenstand der Folge 5. In comFORTH 2 wird diese Information im P-Bit auf Position 6 des Längenbytes im Namenfeld

mitgeführt. Falls es gesetzt ist, handelt es sich um ein bevorzugtes Wort. Das Bit 7 wird durch comFORTH 2 zur Realisierung der Funktionen >NAME und >LINK benutzt. Da innerhalb von Namen nur ASCII-Zeichen im 7-Bit-Code verwendet werden dürfen, ist es möglich, bei der Ermittlung der Namenfeldadresse von der Codefeldadresse aus das Längenbyte durch das gesetzte Bit 7 zu identifizieren. Das Bit 5 ist für Systembenutzung reserviert, wird aber gegenwärtig nicht verwendet. Besonders in Entwicklungsumgebungen muß häufig das Problem gelöst werden, den Namen eines durch seine Namenfeldadresse gegebenen Wortes zur Anzeige zu bringen. Bei einem Kopfaufbau nach Bild 4.3 ist das zum Beispiel so möglich:

```
: .NAME      ( 'nf ===> )
  COUNT 31 AND    TYPE    ;
```

Da der Name in voller Länge gespeichert wird, ist nach Maskierung der höherwertigen Bits des Längenbytes (mittels 31 = 1 FH) direkt das Wort **TYPE** anwendbar. Die Kenntnis der genannten Implementierungsdetails ist bei Anwendung der nachfolgend beschriebenen, standardisierten Suchworte nicht erforderlich. Ein für die Dialogarbeit sehr wichtiges Wort ist das Häkchen (der Apostroph) mit dem Zeichen ' (Aussprache: *tick*). Es ermittelt die Codefeldadresse des im Eingabestrom folgenden Wortes. Als Beispiel dient folgende Kommandozeile:

' DROP >NAME .NAME (cr) DROP ok

Das Häkchen ermittelt hier die Codefeldadresse des Wortes **DROP**. Durch >NAME wird sie in die Namenfeldadresse gewandelt und durch das oben definierte **.NAME** der Name angezeigt. Das Häkchen besitzt bei seiner Verwendung in Anwenderdefinitionen einen Nachteil. Falls die gegebene Zeichenkette nicht in der aktuellen Suchordnung gefunden werden kann, bricht die Suche mit der Fehlermeldung *unbekannt* das aufrufende Wort oder die bearbeitete Kommandozeile mit einer Fehlermitteilung ab. Falls dies nicht erwünscht ist, muß das elementare Suchwort **FIND** verwendet werden, das in der aktuellen Wörterbuchordnung nach einem Wortnamen sucht. FIND liefert als Ergebnis zusätzlich zur Codefeldadresse einen als Flag verwendbaren Parameter. Dieser ist ungleich Null, falls das Wort gefunden wurde, gleich −1, falls es sich um ein einfaches oder gleich 1, falls es sich um ein bevorzugtes Wort handelt.

4.1.3. Indirekter Wortaufruf

Im interpretierenden Systemzustand veranlaßt der Textinterpreter sofort die Abarbeitung eines jeden Wortes, das er mit Hilfe von FIND in der Suchordnung gefunden hat. Dazu benutzt er das Wort **EXECUTE**, dem als Eingangsparameter die Codefeldadresse des auszuführenden Wortes übermittelt wird. EXECUTE und ' sind in ihrem Verhalten aufeinander abgestimmt. Im folgenden Beispiel wird – zu Demonstrationszwecken über den Umweg ' und EXECUTE – das Wort . ausgeführt:

123 ' . EXECUTE (cr) 123 ok

Die Möglichkeit zum Aufruf von Worten über EXECUTE wird gern benutzt, um das Verhalten von Programmen auch nach ihrer Übersetzung noch modifizieren zu können. Im folgenden kleinen Beispiel soll diese als indirekte oder vektorisierte Ausführung bekannte Technik vorgestellt werden. Angenommen, in einem Programm soll es möglich sein, den Stil der Fehlerausschriften zu wechseln. Dazu kann eine

VARIABLE FEHLERTEXT

definiert werden, die die Codefeldadresse desjenigen Wortes enthalten soll, das den Text ausgibt. Zur Auswahl stehen die Worte

```
:  ERNST     ."Fehler"   ;
:  LUSTIG    ."Pech gehabt"   ;
```

Mit Hilfe des Häkchens kann der Zeigervariablen die Codefeldadresse zum Beispiel des „ernsten" Textes zugewiesen werden:

' ERNST FEHLERTEXT !

Zur konkreten Meldungsausgabe wird ein Wort definiert, das die Codefeldadresse aus der Variablen ausliest und mit Hilfe von EXECUTE den aktuellen Text zur Ausgabe bringt:

: MELDUNG FEHLERTEXT @ EXECUTE ;

Zunächst bringt der Aufruf von MELDUNG den oben eingetragenen „ernsten" Text:

MELDUNG (cr) Fehler ok

Durch Einspeichern der Codefeldadresse für den „lustigen" Text kann das Verhalten von **MELDUNG** zu jedem Zeitpunkt verändert werden. Es ist auch möglich, zum Eintragen der entsprechenden Codefeldadressen spezielle Worte zu definieren.

```
:  LUSTIGE  [']  LUSTIG  FEHLERTEXT  !  ;
:  ERNSTE   [']  ERNST   FEHLERTEXT  !  ;
```

Hier wird mit dem Wort **[']** ein Verwandter des Häkchens benutzt. Dieses Wort arbeitet genauso wie ', nur daß es bereits zur Übersetzungszeit ausgeführt wird. Die Kommandofolge ['] **LUSTIG** führt bei Ausführung des Wortes LUSTIGE zur Übergabe der Codefeldadresse von LUSTIG auf dem Parameterstapel. Die beiden oben definierten Worte können folgendermaßen benutzt werden:

LUSTIGE MELDUNG (cr) Pech gehabt ok
ERNSTE MELDUNG (cr) Fehler ok

4.2. Der virtuelle Forthprozessor

Während im Abschnitt 4.1. die Dialogarbeit im Vordergrund stand, wird im weiteren die interne Arbeitsweise eines Forthsystems beschrieben. Hier liegt auch der Schlüssel für die hohe Portabilität der Forthumgebung.

In der Schichtarchitektur eines Forthsystems befindet sich unmittelbar über der Hardware eine Schicht, die virtuelle Maschine genannt wird. In der Tat modelliert diese, im Code des Wirtsrechners programmierte Schicht eine sehr einfache Rechnerkonfiguration. Bild 4.4 gibt einen Überblick über deren Aufbau.

Das Kernstück dieses Rechners ist die CPU, der virtuelle Forthprozessor. Diesem stehen mehrere unabhängige Speicherbereiche zur Verfügung. Da sind der Parameterstapel als Arbeitsspeicher, ein Rückkehrstapel für Organisationszwecke sowie der Hauptspeicher zur Aufnahme von Programmen und Daten. Der Befehlssatz dieses Prozessors ist nicht starr begrenzt, sondern kann erweitert werden. Das erfolgt in der Maschinensprache der verwendeten Hardware. Dieses Verfahren ist am ehesten mit der Mikroprogrammierung vergleichbar. Aus der Sicht des Forthsystems ist es deshalb zweckmäßig, den Maschinencode des Wirtsrechners als *Mikrocode* zu bezeichnen. Der Forthprozessor besitzt einen sehr eingeschränkten Registersatz (siehe Bild 4.5). Er benötigt neben den drei für die Ablaufsteuerung erforderlichen Registern *MIP*, *IP* und *WA* nur zwei Zeigerregister *PSP* und *RSP* für die beiden Stapelspeicher. Die herkömmlichen Allzweckregister werden nicht benötigt, da fast alle Befehle ausschließlich mit Stapeladressierung arbeiten.

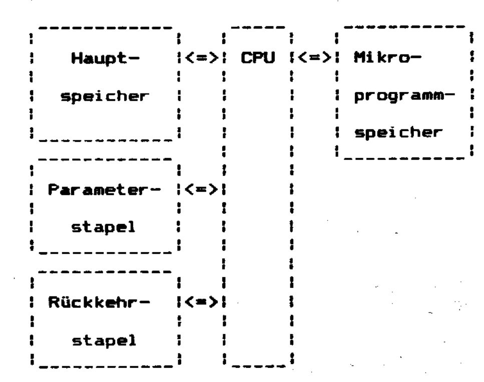


Bild 4.4 Virtuelle Forthmaschine

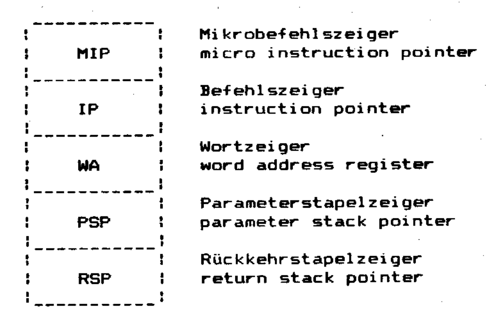


Bild 4.5 Registersatz des virtuellen Forthprozessors

4.2.1. Der Fadencodeaufbau

Der Forthprozessor besitzt eine eigene Maschinensprache, die auf eine Technik zurückzuführen ist, die als Fadencode bekannt wurde. Fadencode wird durch den Forthübersetzer während der Kompilation von Doppelpunktdefinitionen erzeugt. Für die Implementierung von Fadencode gibt es verschiedene, qualitativ jedoch gleichwertige Varianten. Am häufigsten wird die doppelt indirekte Technik verwendet, auf die sich die weiteren Aussagen beziehen.

Bei dieser Fadencodevariante wird die letztlich aufzurufende Mikrocodepassage in zwei Stufen indirekt verschlüsselt. Die erste Stufe ist der Fadencode selbst, der aus einer Aufzählung der Codefeldadressen der nacheinander aufzurufenden Worte besteht. Die zweite Verschlüsselung erfolgt auf den Codefeldern der Worte. Diese enthalten die Adresse des Mikroprogramms, das bei Aufruf des jeweiligen Wortes auszuführen ist. Diese Mikroprogramme organisieren die Behandlung der Parameterfelder, deren Aufbau von der Art des Wortes abhängt. Auf diese Weise

können alle Wortarten mit derselben Methode aufgerufen werden. Bild 4.6 zeigt den Aufbau des Fadencodes, der bei einer Definition

: TUCK SWAP OVER ;

erzeugt würde. Hier wird angenommen, daß sowohl **SWAP** als auch **OVER** Mikroprogramme sind. Deshalb verweisen deren Codefelder jeweils auf die eigenen Parameterfelder, die den zugehörigen Mikrocode enthalten. Solche Worte werden auch Primärworte genannt.

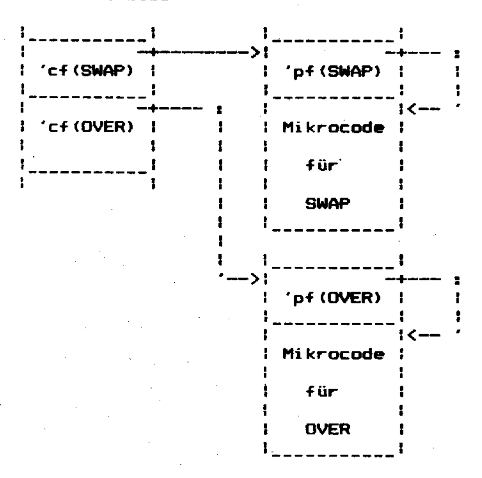


Bild 4.6 Beispiel für Forthcode

4.2.2. Der Adreßinterpreter

Die Abarbeitung von Fadencode erfolgt durch die Ablaufsteuerung des virtuellen Forthprozessors, die aufgrund ihrer Arbeitsweise oft Adreßinterpreter genannt wird. Da sie die Ausführung der jeweils nächsten Anweisung organisiert, besitzt die Einsprungstelle meist den Namen **NEXT**. Diese Ablaufsteuerung benötigt zu ihrer Arbeit drei Register (siehe Bild 4.5). Die Funktion des Adreßinterpreters kann in der Sprache eines fiktiven Prozessors folgendermaßen beschrieben werden:

```
NEXT: LD   WA,(IP)   ;WA zeigt auf 'cf
      INC  IP        ;IP weiter setzen
      LD   MIP,(WA)  ;Mikroprogramm aus
                     ;Codefeld anspringen
```

Zentrale Bedeutung für die Ablaufsteuerung hat der Befehlszeiger **IP**. Beim Eintritt in NEXT muß er auf die nächste zu bearbeitende Fadencodeanweisung zeigen. Dann wird zuerst die Codefeldadresse des aufzurufenden Worts in das *WA*-Register gebracht und der Befehlszeiger auf die folgende Fadencodeanweisung vorgerückt. Anschließend wird der Mikrobefehlszähler (also der Befehlszähler der Wirtsmaschine) mit derjenigen Adresse geladen, die auf dem Codefeld des abzuarbeitenden Wortes steht. Damit wird die Steuerung an das entsprechende Mikroprogramm abgegeben, das seinerseits unbedingt mit einem Sprung zurück zu NEXT enden muß. Bild 4.7 zeigt eine typische Implementierung für den Prozessor U 880.

```
; --------- Registerbelegung ----------

; Befehlszeiger               - BC
; Wortzeiger                  - DE
; Parameterstapelzeiger       - SP
; Rückkehrstapelzeiger        - Speicherzelle
; Mikroprogrammzeiger         - PC

NEXT: LD    A,(BC)    ;lesen low('cf)
      INC   BC        ;IP ein Byte weiter
      LD    L,A       ;HL ist WA
      LD    A,(BC)    ;high('cf) analog
      INC   BC        ;nächster Befehl
      LD    H,A       ;'cf vollständig
      LD    E,(HL)    ;DE ist Zwischen-
      INC   HL        ;speicher für MIP
      LD    D,(HL)    ;WA:='cf+1 !!!
      EX    DE,HL     ;DE wird WA
      JP    (HL)      ;laden MP
```

Bild 4.7 Ablaufsteuerung für den Prozessor U 880

Der Fadencode besteht nicht ausschließlich aus einer Aneinanderreihung von Codefeldadressen. Verschiedenen Worten werden auch mit Hilfe der unmittelbaren Adressierung Parameter übermittelt. Zu diesen Worten gehören zum Beispiel die im Systemerweiterungswortschatz standardisierten Worte **BRANCH** und **?BRANCH**, mit denen bedingte und unbedingte Sprünge innerhalb des Fadencodes realisiert werden. Bei ihnen findet man unmittelbar im Fadencode folgend das Sprungziel (Bild 4.8). In der Sprache des fiktiven Prozessors formuliert ruft BRANCH das folgende Mikroprogramm auf:

```
BRANCH: LD  IP,(IP)    ;laden Zieladresse
        LD  MIP,NEXT   ;Rückkehr zur
                       ;Ablaufsteuerung
```

BRANCH und ?BRANCH werden beispielsweise durch die Worte **IF** und **ELSE** benutzt. Einzelheiten dazu sind Gegenstand der nächsten Folge.

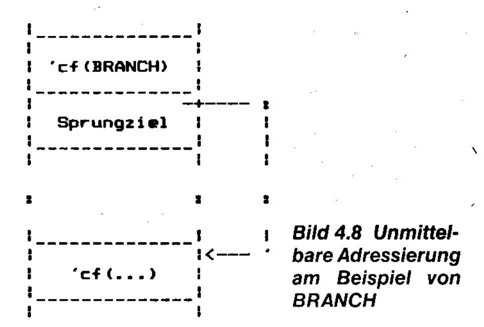


Bild 4.8 Unmittelbare Adressierung am Beispiel von BRANCH

4.2.3. Aufruf von Fadencodeprogrammen

Eines der wesentlichen Merkmale von Fadencode ist, daß die Art des Wortes durch den Inhalt seines Codefeldes verschlüsselt ist. Damit kann der Aufruf von Primärworten genauso erfolgen, wie der von Doppelpunktworten (Sekundärworten). Dafür verantwortlich ist das durch diese Worte aufgerufene Mikroprogramm mit dem Namen **DOCOL** (von englisch *do-colon*). Es entspricht der CALL-Anweisung herkömmlicher Prozessoren:

```
DOCOL: DEC  RSP        ;Platz auf RS
       LD   (RSP),IP   ;retten IP
       INC  WA         ;WA: = 'pf
       LD   IP,(WA)    ;IP auf Programm
       LD   MIP,NEXT   ;Ablaufsteuerung
```

Durch DOCOL wird der momentane Stand des Befehlszeigers *IP*, also die Fortsetzungsadresse, auf den Rückkehrstapel gerettet.

Da sich das aufzurufende Fadencodeprogramm auf dem Parameterfeld des gerufenen Wortes befindet, wird dessen Adresse in den Befehlszeiger eingetragen. Die Rückkehr aus einem so aufgerufenen Fadencodeprogramm wird durch das am Ende jedes Fadencodeprogramms kompilierte Primärwort **EXIT** organisiert.

```
EXIT: LD   IP,(RSP)   ;Fortsetzungsadresse
      INC  RSP        ;Platz freigeben
      LD   MIP,NEXT   ;Ablaufsteuerung
```

EXIT entspricht der Return-Anweisung üblicher Prozessoren. Das Zusammenspiel von DOCOL und EXIT soll anhand des Aufrufs des oben definierten Wortes **TUCK** aus dem Wort

```
: 2!     ( ps: 32b addr ===> )
  TUCK   ! 2+ ! ;
```

illustriert werden. Bild 4.9 zeigt den Aufbau des zugehörigen Fadencodes. Die Adresse X und Y bezeichnen die Parameterfeldadresse von 2! bzw. TUCK.

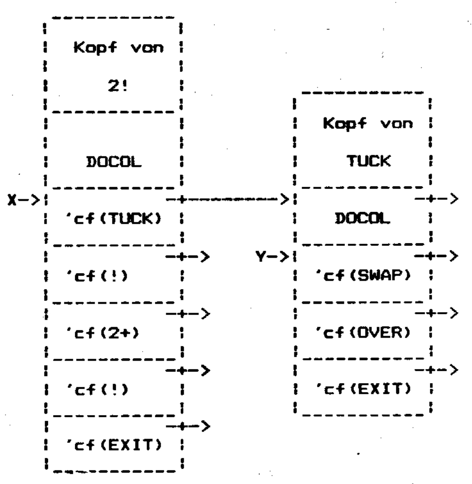


Bild 4.9 Interne Repräsentation der Worte TUCK und 2!

In der Tafel 4.1 kann die Registerbelegung und die Stapelbelegung während der Abarbeitung von **2!** verfolgt werden. Als Ausgangssituation wird der Aufruf von 2! aus einem Fadencodeprogramm auf der Adresse *ip* angenommen. In der ersten Spalte sind die durch die Ablaufsteuerung nacheinander aufgerufenen Mikroprogramme, in den weiteren der Zustand nach ihrer Ausführung aufgeführt. Die als *pn* bzw. *rn* aufgeführten Parameter werden nicht beeinflußt.

Tafel 4.1 Abarbeitung des Wortes 2!

Mikro-programm	Parameterstapel				Rückkehrstapel		IP
	3	2	1	0	1	0	
...	p3	1(32b)	h(32b)	addr	r1	r0	ip
DOCOL	p3	1(32b)	h(32b)	addr	r0	ip+2	X
DOCOL	p3	1(32b)	h(32b)	addr	ip+2	X+2	Y
SWAP	p3	1(32b)	addr	h(33b)	ip+2	X+2	Y+2
OVER	1(32b)	addr	h(32b)	addr	ip+2	X+2	Y+4
EXIT	1(32b)	addr	h(32b)	addr	r0	ip+2	X+2
!	p4	p3	1(32b)	addr	r0	ip+2	X+4
2+	p4	p3	1(32b)	addr+2	r0	ip+2	X+6
!	p6	p5	p4	p3	r0	ip+2	X+8
EXIT	p6	p5	p4	p3	rl	r0	ip+2

4.2.4. Der Rückkehrstapel

Der Rückkehrstapel darf mit Vorsicht auch für das zwischenzeitliche Speichern von Daten verwendet werden. Zu diesem Zweck bietet der Standard drei Funktionen an, die den Transfer von Daten zwischen dem Parameterstapel und dem Rückkehrstapel ermöglichen. Dies sind:

```
>R  ps:  16b ===> rs: ===> 16b
R>  ps:  ===> 16b rs: 16b ===>
R@  ps:  ===> 16b rs: 16b ===> 16b
```

Bei der Benutzung dieser Befehle ist zu beachten, daß der Rückkehrstapel durch die Ablaufsteuerung benutzt wird. Sie sollten deshalb nur innerhalb einer Aufrufebene verwendet werden. Es ist darauf zu achten, daß vor dem Rücksprung in die aufrufende Programmebene alle Parameter wieder entfernt werden. Diese Vorgehensweise entspricht der Benutzung des Prozessorstapels bei Assemblerprogrammierung. Trotz dieser Einschränkungen bleibt der Rückkehrstapel ein wichtiges Hilfsmittel zur Vermeidung umfangreicher Stapelmanipulationen. Als Beispiel dafür kann die Hochwortdefinition des Wortes **2SWAP** herangezogen werden.

```
: 2SWAP ( ps: 32b ===> 32b )
  ROT >R ROT R> ;
```

Ohne Verwendung des Rückkehrstapels hätte man die wesentlich langsamere Variante

```
: 2SWAP   3 ROLL   3 ROLL   ;
```

verwenden müssen.

Mit Hilfe des Rückkehrstapels ist es auch möglich, Fadencodeprogramme indirekt anzuspringen und auszuführen. Dazu braucht nur deren Adresse auf den Rückkehrstapel gelegt werden. Beim nächsten Aufruf von EXIT wird diese dann in das *IP*-Register geladen und das zugehörige Programm bearbeitet. (Dieses Verfahren ist auch in der Assemblerprogrammierung bekannt: Eine Zieladresse wird dadurch angesprungen, daß sie auf den Stapel gelegt wird und daß danach ein Returnbefehl ausgeführt wird.) Die (nicht interaktiv ausführbare) Kommandofolge

'. >BODY >R

führt in diesem Sinne zum Aufrufen des Fadencodes der Zahlenausgabe. Diese Methode ist auf Fadencode beschränkt und benötigt nicht die Existenz eines Codefeldes.

4.3. Mikroprogrammierung

Bei der Programmierung prozeßnaher Anwendungen taucht immer wieder das Problem auf, Teile des Gesamtprogramms im Maschinencode des Hardwareprozessors formulieren zu müssen. Dies betrifft insbesondere sehr zeitkritische Abschnitte oder die Bedienung spezieller Peripherie. In einem Forthsystem ist solcher Code sehr organisch in Form neuer Mikroprogramme des Forthprozessors integrierbar.

4.3.1. Assembler in Forth

Für das Erzeugen von Primärworten wird durch den Standard im Erweiterungswortschatz für Assemblerprogrammierung die Konstruktion

CODE name ... END-CODE

definiert. Ihre Benutzung erfolgt analog zur Konstruktion

: name ... ;

zur Definition von Fadencodeprogrammen. Anstelle der Aufzählung von Forthworten erfolgt die Angabe der Assemblerbefehle, die nacheinander ausgeführt werden sollen. Im Gegensatz zur Doppelpunktkonstruktion verbleibt das System zwischen **CODE** und **END-CODE** im Ausführmodus.

Die unter Regie des Forthsystems benutzten Assemblerbefehle sind wie alle anderen Kommandos ebenfalls in Form von Forthworten verfügbar. Sie werden in dem Vokabular **ASSEMBLER** aufbewahrt, das durch CODE automatisch in die Wörterbuch-Suchordnung aufgenommen und durch END-CODE wieder aus dieser entfernt wird.

Die Worte des Assemblervokabulars unterliegen keiner Standardisierung. Es haben sich jedoch einige Konventionen durchgesetzt, die im weiteren beschrieben werden. Entsprechend der Systemphilosophie benutzt dieser Assembler die umgekehrt polnische Notation. Das bedeutet, daß zuerst die Operanden, zum Beispiel Registernamen, und danach die Operationen angegeben werden. Bei Zweioperandenbefehlen wird analog zum Wort ! zuerst der Quelloperand und danach der Zieloperand erwartet. Zur Unterscheidung der Mnemoniks von standardisierten Worten wie **AND** oder **OR** enden alle Assemblerworte mit einem Komma. Da das System im Ausführmodus verbleibt können Zahlen als Direktoperanden einfach über den Stapel übergeben werden. Dabei ist es belanglos, ob sie direkt als Zahl eingegeben werden oder das Ergebnis einer Berechnung sind. Zur Sicherung der Syntax werden sie mit dem Wort # markiert, das im Assemblervokabular eine spezielle Bedeutung besitzt. Die Tafel 4.2 zeigt dies am Beispiel des Additonsbefehls des U 880.

Tafel 4.2 Forth-Assemblernotation

symbolische Notation	Original-Notation	FORTH-Notation
ADD r	ADD L	L ADD,
ADD n	ADD 12	12 # ADD,
ADD ii+d	ADD (IX+7)	7 (IX) ADD,
ADD HL,dd	ADD HL,DE	DE HL ADD,

4.3.2. Programmierbeispiel

Als Abschluß dieses Abschnitts soll anhand der Implementierung des oben mit Hilfe des Doppelpunkts definierten Wortes TUCK ein Beispiel für den Umgang mit dem U 880-Assembler vorgeführt werden.

Der in Bild 4.10 zu findende Quelltext bezieht sich auf die in Bild 4.7 gezeigte Ablaufsteuerung für den U 880. Diese verwendet das Register *BC* als Befehlszeiger. *BC* darf deshalb entweder durch den Assemblertext nicht verändert werden, oder sein Inhalt muß wieder rekonstruiert werden. Beim Code von TUCK wurde dies berücksichtigt.

```
CODE TUCK     ( ps: 16b1 16b2 ===>       )
              (           16b2 16b1 16b2 )
     HL POP,         ( lesen 16b2        )
     DE POP,         ( lesen 16b1        )
     HL PUSH,        ( schreiben 16b2    )
     DE PUSH,        ( schreiben 16b1    )
     HL PUSH,        ( schreiben 16b2    )
     NEXT # JP,      ( nächster Befehl )
     END-CODE
```

Bild 4.10 ***Assemblerimplementierung von TUCK***

Im Gegensatz dazu darf das Register *DE*, das als Wortzeiger benutzt wird, zerstört werden, da es immer wieder neu gesetzt wird. Das Sprungziel NEXT sollte der Assembler als symbolische Bezeichnung bereitstellen.

wird fortgesetzt

Tafel 4.3 Kurzbeschreibung der Forthworte

Name	Stapeleffekt	Beschreibung
Wortstruktur		
>BODY	('cf ===> 'pf)	Umrechnung der Code- in die Parameterfeldadresse
>LINK	('cf ===> 'lf)	Umrechnung der Code- in die Verkettungsfeldadresse
>NAME	('cf ===> 'nf)	Umrechnung der Code- in die Namensfeldadresse
BODY>	('pf ===> 'cf)	Umrechnung der Parameter- in die Codefeldadresse
L>NAME	('lf ===> 'nf)	Umrechnung der Verkettungs- in die Namensfeldadresse
LINK>	('lf ===> 'cf)	Umrechnung der Verkettungs- in die Codefeldadresse
N>LINK	('nf ===> 'lf)	Umrechnung der Namens- in die Verkettungsfeldadresse
NAME>	('nf ===> 'cf)	Umrechnung der Namens- in die Codefeldadresse
Suchen und indirekter Aufruf		
'	(===> 'cf) ib: name	Ermittelt die Codefeldadresse des Worts, dessen Name im Eingabepuffer folgt (ib – input buffer).
[']	(===>) ib: name	Analog zu ', nur zur Übersetzungszeit. Die Codefeldadresse wird als Literal gespeichert und zur Laufzeit auf dem Stapel übergeben.
EXECUTE	('cf ===>)	Das durch die Codefeldadresse gegebene Wort wird ausgeführt.
FIND	(addr ===> 'cf n) oder (addr ===> addr 0)	Die Kette ab addr wird im Wörterbuch gesucht. Falls ein Wort dieses Namens existiert, wird addr durch die Codefeldadresse ersetzt, n ist 1 für bevorzugte Worte, sonst –1. Falls das Wort nicht gefunden wird, liegt über addr *falsch*.
Virtueller Prozessor		
>R	(16b ===>) rs: ===> 16b	Transport von 16b auf den Rückkehrstapel
R>	(===> 16b) rs: 16b ===>	Rücktransport von 16b vom Rückkehrstapel
R@	(===> 16b) rs: 16b ===> 16b	Kopie der Rückkehrstapelspitze
BRANCH	(===>)	Unbedingter Sprung, das Ziel wird im Fadencode übergeben
?BRANCH	(? ===>)	Sprung falls Null, sonst wie **BRANCH**
Mikroprogrammierung		
ASSEMBLER	(===>)	Vokabular zur Aufnahme des Assemblerwortschatzes
CODE	ib: name	Definionswort für ein Primärwort, dessen Name im Eingabepuffer folgt. Muß durch **END-CODE** abgeschlossen werden.
END-CODE	(===>)	Abschluß einer Primärdefinition

Interrupt und Einzelschritt in Forth

Peter Taege, Berlin

Verschiedentlich wird in Forth auch die an sich nicht zum Standard gehörende Interruptverarbeitung vorgesehen /1/. In der hier beschriebenen Variante wird von nur einer Interruptquelle ausgegangen.
Zur Lösung des Problems muß eine Verbindung zwischen Prozessorebene und Forth geschaffen werden. Dazu wurden drei Verbindungszellen mit folgenden Aufgaben eingeführt:

- IF1 steuert die Interruptannahme
- in IF2 wird die Information über einen Prozessorinterrupt an Forth übergeben
- INTAD enthält die CFA des Wortes, das bei Interrupt abgearbeitet werden soll.

In der Interruptserviceroutine (ISR) für die vorgesehene Interruptquelle (z. B. PIO oder CTC) fragt der Prozessor Bit 7 in IF1 ab. Ist es gesetzt, also bei erlaubtem Interrupt, lädt er IF2 mit dem Wert 40H (Bild 1), anderenfalls geht der Interrupt verloren. IF2 wird im Forth-System am Anfang der NEXT-Routine ausgewertet (Bild 2). Durch das Schieben in IF2 entsteht eine Verzögerung, die garantiert, daß das Wort EXIT am Ende der Forth-ISR noch durchlaufen wird, bevor ein erneuter Interrupt wirksam werden kann.
Bei angenommenem Interrupt lädt der Prozessor das Register HL mit der in INTAD stehenden CFA der Forth-Interruptroutine und schiebt diese so in den Ablauf ein, als wäre

```
M_ISR:  PUSH AF
        LD A,(IF1+2)     ;PFA von IF1
        BIT 7,A
        JR Z,ISREND
          LD A,40H
          LD (IF2+2),A
ISREND: POP AF
        EI
        RETI
```

Bild 1 Prozessor-ISR

```
NEXT: LD HL,IF2+2          ;PFA von IF2
      SLA (HL)
      JR NC,NOINT
        LD HL,IF1+2        ;Interrupt angenommen
        RES 7,(HL)         ;DI
        LD HL,(INTAD+2)    ;CFA der FORTH-ISR
      JR EXEC1
NOINT:  .                 ;Fortsetzung von
        .                 ;NEXT
        .
EXEC1:LD E,(HL)
      INC HL
      LD D,(HL)
      EX DE,HL
      JP (HL)
```

Bild 2 Erweiterte NEXT-Routine

sie Bestandteil des im Hintergrund laufenden Programms. Die Variable INTAD ist mit der gewünschten Adresse zu füllen, bevor der Interrupt erstmalig erlaubt wird.
Wie aus Bild 2 zu ersehen ist, wird der Interrupt nach einer Annahme verboten. Er muß daher innerhalb der Forth-ISR durch das zusätzlich definierte Wort F_EI wieder erlaubt werden. Als Pendant dazu wurde das Wort F_DI gebildet, welches das Bit 7 in IF1 zurücksetzt.
Nach Einfügung von F_EI kann jedes Forth-Wort Interruptroutine sein, sofern es die Stacks in geordnetem Zustand hinterläßt. Nur dann ist der ordnungsgemäße Ablauf des Hintergrundprogramms gesichert. Ansonsten können Parameter- und Returnstack ohne Einschränkung benutzt werden.
Prozessorinterrupts werden auf Forth-Ebene beim darauffolgenden Sprung nach NEXT, also nach Abschluß des laufenden Maschinenkodeprogramms, wirksam. So wie der Prozessor nach einem Interrupt einen Befehl noch bis zu Ende bearbeitet, werden in Forth die laufenden elementaren Maschinenkoderoutinen abgeschlossen.
Die ungünstigste Interruptreaktionszeit ist also gleich der Laufzeit des längsten im System vorhandenen oder durch Erweiterung hinzugefügten Maschinenprogramms.
Da alle Verbindungszellen als Forth-Worte deklariert wurden, besteht die Möglichkeit, einen Software-Interrupt auszulösen. Damit ist mit wenig zusätzlichem Aufwand ein Einzelschrittbetrieb realisierbar.

```
HEX
: RET CR R@  @ DUP U. ." : "   (IP-Stand)
      >R .S R> CR              (Par.-Stack)
      @ 2+ NFA COUNT TYPE      (Name nächstes Wort)
      KEY DROP 20 IF2 C! ;

: STEP CFA 20 IF2 C!           (Start Soft-Interrupt)
      EXECUTE
      STEPEND ;                (Stop  Soft-Interrupt)
```

Bild 3 Worte für Einzelschritt

Dabei wird ausgenutzt, daß die erweiterte NEXT-Routine die Möglichkeit bietet, in den Programmablauf andere Worte einzuschieben. Für den Einzelschrittbetrieb ist dies das Programm RET (Bild 3), welches in der hier dargestellten einfachen Variante den Instruktionspointer, den Namen des nächsten Wortes und den Parameterstack anzeigt. Es kann natürlich beliebig modifiziert werden.
Die Vorbereitung des Systems auf den Schrittbetrieb beginnt mit dem Ablegen der ISR-Adresse nach INTAD:

' RET CFA INTAD !

Anschließend wird die PFA des zu testenden Wortes an STEP übergeben und der Schrittbetrieb gestartet:

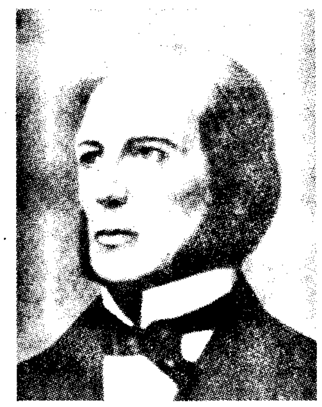
' name STEP

Nun wird nach jedem Schritt das Wort RET angelaufen. Nach den oben genannten Anzeigen wird der Ablauf durch KEY gebremst und nach Tastendruck fortgesetzt. Am Ende von RET wird IF2 zur Auslösung des nächsten Software-Interrupts mit 20H geladen.
Die Kette der sich immer wieder selbst aktivierenden Software-Interrupts wird durch das Wort STEPEND unterbrochen. Es ist in Maschinenkode geschrieben und setzt IF2 auf 0. Sein Name wird im Schrittbetrieb noch angezeigt, bevor das System zur QUIT-Grundroutine zurückkehrt.

Literatur

/1/ Briner, R., G.: Ein- und Ausgabe in Forth. Elektroniker Nr. 5/1986, S. 86

Wegbereiter der Informatik



GEORGE BOOLE

∗ 1815 Lincoln (England),
† 1864 Ballintemple (Irland)

G. Boole hat eine Algebra erfunden, die mit Elementen operiert, welche nur die beiden Werte 0 und 1 annehmen. Als Verknüpfungen sind dabei die binären Operationen Konjunktion und Disjunktion sowie die Komplementbildung (Negation) erklärt. Diese Algebra ist letztlich aus dem Bestreben Booles entstanden, logische Aussagen so darzustellen, daß diese nicht durch die oftmals zweideutige Umgangssprache verfälscht werden können. Er wurde damit zum Mitbegründer der mathematischen Logik. Auch Leibniz hatte übrigens diesen Gedanken schon gefaßt und als Zwanzigjähriger mit seiner darauf bezogenen Schrift *Dissertatio de arte combinatoria* 1666 die Lehrbefugnis an der Leipziger Universität erhalten.
Aus der *Booleschen Algebra* ist als Anwendung die *Schaltalgebra* hervorgegangen, ohne die die heutige Computertechnik kaum denkbar wäre, denn sie bietet grundsätzliche Möglichkeiten für die Anwendung der Logik in der Elektrotechnik.
Boole war Autodidakt und ist Zeit seines Lebens als Außenseiter angesehen worden. Noch fünfzig Jahre nach seinem Tode hielten viele seine Theorie für philosophische Spielerei und lehnten es ab, sie als Mathematik anzuerkennen.
Nach Besuch einer Handelsschule wurde er noch im Jünglingsalter Hilfslehrer. Nebenher studierte er mathematische Schriften von Newton sowie die Mechanik von Lagrange. Er fand bald heraus, daß die Variablen x, y, ... in algebraischen Relationen wie

$$x + y = y + x$$

nicht unbedingt Zahlen repräsentieren müssen, sondern auch durch andere Begriffe (oder Aussagen) belegt werden können. So erarbeitete er Gesetzmäßigkeiten des logischen Schließens, und 1848 erschien seine erste diesbezügliche Publikation: *The Mathematical Analysis of Logic*. Ein Jahr darauf wurde Boole, der nie ein Studium an einer Hochschule absolviert hatte, als Professor für Mathematik an das neu gegründete Queens College in Cork (Irland) berufen. Bereits 1854 ist sein Hauptwerk *An Investigation of the Laws of Thought* veröffentlicht worden. Darin wird die anfangs erwähnte Boolesche Algebra ausführlich behandelt. Sie liefert übrigens mittelbar (über die *Booleschen Funktionen*) auch Regeln zur Aufstellung von Strukturmustern in kybernetischen Systemen.
Zusammen mit der kongenialen Leibnizschen Entdeckung des dualen Zahlensystems waren nunmehr – zumindest theoretisch – wesentliche Voraussetzungen geschaffen, um die mathematischen Grundoperationen im Computer realisieren zu können. Die erste vollständige technische Verwirklichung erfolgte allerdings erst 1941 durch Konrad Zuse.

Dr. Klaus Biener

Mittel und Methoden der Künstlichen Intelligenz Teil 2

Jörg Schmidt
Technische Universität Karl-Marx-Stadt, Sektion Informatik

Suchstrategien

Die nachfolgend beschriebenen Suchstrategien finden eine Folge von anzuwendenden Regeln (einen Weg), die aus einer gegebenen Datenbasis eine Zieldatenbasis ableitet. Dieser Weg muß nicht immer der kürzeste und optimale Weg sein. Oftmals ist es auch schon entscheidend, ob dieser Weg existiert oder nicht. Bei einer Reihe von kombinatorischen Aufgaben ist aber gerade der optimale Weg angestrebt, um das Problem überhaupt in akzeptabler Rechenzeit lösen zu können.

Backtracking

Beim Backtracking werden die Regeln nacheinander ausgewählt und zur Anwendung gebracht. Sollte der durch die Folge dieser Regeln bestimmte Lösungsweg fehlschlagen, das heißt zu einem Widerspruch führen (beispielsweise in Prolog, wenn zwei Prädikate sich nicht gegeneinander matchen lassen), oder zu lang werden oder sehr weit von der angestrebten Lösung entfernt sein, wird das weitere Verfolgen dieses Lösungsweges aufgegeben, und man geht zu einem Punkt, an dem alternative Möglichkeiten bestehen, zurück, von wo an eine andere Folge von Regeln angewandt wird.
Nachfolgend wird eine einfache rekursive Prozedur (Funktion) beschrieben, die das Backtracking-Verfahren zeigt. Diese Prozedur wird auf eine Datenbasis (DB) angewandt und hat DB als Argument. Um das Verständnis zu erhöhen, werden die einzelnen Schritte verbal beschrieben.

BACKTRACKING (DB)

(1) Wenn die Datenbasis (DB) die Endebedingung erfüllt, gib die leere Liste zurück; denn dann sind keine Regeln weiter anzuwenden.

(2) Wenn sicher ist, daß DB nicht auf dem Weg zur Lösung liegt, gib FAIL für *fehlgeschlagen* zurück.

(3) Ermittle alle Regeln, die auf DB angewandt werden können. Diese Regeln bilden die Liste REGELN.

(4) SCHLEIFENANFANG:
Wenn die Anzahl der Elemente in REGELN gleich Null ist, dann gibt es keine Regeln mehr, die auf DB angewandt werden können, und FAIL wird zurückgegeben.

(5) Die erste Regel R von REGELN wird von der Liste gelöst und zur Anwendung auf DB ausgewählt.

(6) REGELN wird der Restlinie von REGELN gleichgesetzt. Damit verringert sich die Anzahl der Regeln in der Liste, so daß es durchaus möglich ist, daß wie im Schritt 4 untersucht, die Anzahl der Elemente der Liste den Wert 0 annimmt.

(7) Die Regel R wird auf die Datenbasis DB angewandt und bildet dadurch eine neue Datenbasis RDB.

(8) BACKTRACKING wird mit dem Argument RDB aufgerufen:
WEG:=BACKTRACKING(RDB).

(9) Wenn der Aufruf der Prozedur WEG:= BACKTRACKING(RDB) fehlschlägt, also WEG=FAIL, dann wird zu SCHLEIFENANFANG zurückgegangen und der Lösungsweg mittels einer anderen Regel, soweit möglich, gesucht.

(10) Wenn BACKTRACKING erfolgreich war (die Bedingung im Schritt 9 also nicht erfüllt), dann wird die ausgewählte Regel R an den Anfang der Liste WEG gesetzt.

Die hier dargestellte Prozedur ist ein sehr vereinfachtes BACKTRACKING-Verfahren. Sie hat eine Reihe von Nachteilen. So ist beispielsweise die Tiefe des Suchverfahrens, das heißt die Länge des Lösungsweges, nicht beschränkt. Damit ist es durchaus auch wahrscheinlich, daß die Suche sehr weit in die Tiefe getrieben wird und trotzdem nicht zur Lösung führt.
Zur Illustration dient uns folgendes Beispiel. Es sind auf einem 4 × 4-Schachbrett 4 Damen so aufzustellen, daß sie einander nicht schlagen können (Der Einfachheit halber ist das Problem auf 4 Damen und ein 4 × 4-Brett begrenzt). Es dürfen also keine zwei Damen eine gemeinsame Gerade oder eine gemeinsame Diagonale haben.
Regel R(i,j) stellt eine Dame in Reihe i und Spalte j, unter der Bedingung, daß Reihe (i–1) bereits besetzt ist (für i>1).
Entsprechend unserer Prozedur erfüllt das leere Schachbrett die Endbedingung nicht, und im Schritt 3 erhalten wir die Regelliste:
(R(1,1),R(1,2),R(1,3),R(1,4))

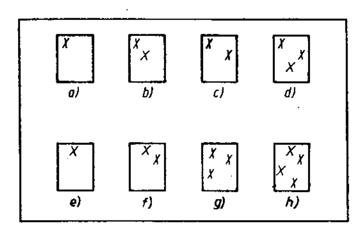


Bild 10 4-Damen-Problem auf einem 4×4-Brett

Wir wählen im Schritt 5 die erste Regel R(1,1) aus und wenden sie an (Bild 10a). Uns verbleiben die Regeln R(1,2), R(1,3), R(1,4) für eine erneute Anwendung, wenn R(1,1) nicht zum Ziel führt. Auf die neue Datenbasis, das Schachbrett mit der ersten besetzten Reihe, wird unsere Prozedur erneut angewandt. Schritt 3 ermittelt R(2,3) und R(2,4) als mögliche Regeln. R(2,3) wird angewandt und ergibt Bild 10b. Beim nächsten Aufruf der Prozedur ist die Anzahl der im Schritt 3 ermittelten Regeln gleich 0. Es wird in (4) das Prädikat FAIL zurückgegeben und in (9) zum Schleifenanfang zurückgegangen.
R(2,4) befindet sich noch als einziges Element in der Liste der anwendbaren Regeln. Wir erhalten Bild 10c. Im nächsten Aufruf der Prozedur berechnet (3) R(3,2) als einzig mögliche Regel und wir erhalten die in Bild 10d dargestellte Konfiguration. Wieder wird unsere Prozedur auf diese Datenbasis angewandt, kann aber im Schritt 3 keine Regel R(4,j) ermitteln und gibt in (4) das Prädikat FAIL zurück. Die Liste der Regeln R(3,j) ist auch bereits leer, und es wird in (4) wieder FAIL zurückgegeben. Auch die Regeln für die zweite Reihe R(2,j) sind erschöpft, und es wird nochmals FAIL zurückgegeben. Wir haben damit wieder die Liste (R(1,2),R(1,3), R(1,4)). Als nächstes wird R(1,2) angewandt und ergibt Bild 10e. Der erneute Aufruf von BACKTRAKKING ermittelt für die zweite Reihe als einzige Regel: R(2,4). Diese wird auf die Datenbasis angewandt. (Bild 10f) Der nächste Aufruf unserer Prozedur ergibt R(3,1) als einzig mögliche Regel (Bild 10g), und BACKTRACKING nochmals aufgerufen ergibt Bild 10h. Diese Konfiguration entspricht auch der Endebedingung. Der erneute Aufruf der Prozedur gibt die leere Liste in Schritt 1 zurück und in (10) werden die einzelnen Regeln nacheinander zum Lösungsweg formiert:

(R(4,3))
(R(3,1),R(4,3))
(R(2,4),R(3,1),R(4,3))
(R(1,2),R(2,4),R(3,1),R(4,3))

Das Backtracking-Verfahren wird unter anderem in der Programmiersprache Prolog angewandt. Wie das Beispiel gezeigt hat, wird nur eine Lösung ermittelt. Prinzipiell ist es möglich, nach der Ausgabe einer Lösung FAIL zu erzwingen, so daß das Backtracking-Verfahren alle möglichen Lösungen ermitteln muß.

Graphensuche

Das Backtracking-Verfahren führt zu einer effektiven Nutzung des Speicherplatzes, der zur Verfügung steht. Lösungswege, die zu FAIL geführt haben, werden wieder vergessen. Das ist beim heutigen Stand der Computertechnik gewiß noch als Vorteil zu sehen. Flexibler ist jedoch ein Verfahren, das sich alles, was irgendwann einmal auf dem Weg zur Lösung lag, merken und gegebenenfalls darauf zurückgreifen kann. Das wird erreicht, indem wir das gesamte Suchverfahren, also auch alle Datenbasen, in einem Graphen darstellen.
Ein Graph besteht damit aus einer Menge von Knoten, die jeweils eine Datenbasis repräsentieren, und einer Menge von gerichteten Kanten zwischen diesen Knoten. Diese widerspiegeln die Anwendung einer Regel, die von einer Datenbasis DB1 zu einer Datenbasis DB 2 führt.
Wenn eine Kante vom Knoten k(i) zum Knoten k(j) führt, dann ist k(j) ein Nachfolgeknoten von k(i) und k(i) ein Elternknoten von k(j). Ein Baum ist ein Graph, in dem jeder Knoten höchstens einen Elternknoten besitzt. Derjenige Knoten, der keinen solchen Elternknoten besitzt, wird als Wurzelknoten bezeichnet. Dem Wurzelknoten wird die Tiefe 0 zugeordnet. Alle anderen Knoten haben eine um 1 erhöhte Tiefe ihres Elternknoten. Soviel zu einigen wesentlichen Grundbegriffen.
Die Aufgabe eines Suchverfahrens auf dieser Grundlage ist das Finden eines Lösungspfades von einem Anfangsknoten a zu einem Element aus einer Menge {e(i)} von Knoten, die die Ende- bzw. Zielbedingung erfüllen. Dazu ist es notwendig, ausgehend von a jeweils entsprechend der möglichen Regelanwendungen die Nachfolgeknoten zu ermitteln, von diesen wieder die Nachfolgeknoten usw. Dabei darf in der Menge der Nachfolgeknoten eines Knotens k(i) kein k(j) auftreten, der auf einem Pfad (k(j)=k(j,0),k(j,1),k(j,2),...k(j,n)=k(i)) zu k(i) geführt hat.

Entscheidend bei diesem Suchverfahren ist die Frage, welcher der Nachfolgeknoten zuerst (oder überhaupt) zur Expansion ausgewählt wird.
Wir unterscheiden daher blinde Suchverfahren, die keine Information über das Problem besitzen, und heuristische Verfahren. Zu den ersteren zählen die Verfahren Tiefe-Zuerst (Tiefensuche) und Breite-Zuerst (Breitensuche).

Tiefensuche
Aus der Menge der Knoten des Graphen, deren Nachfolger ermittelt werden können, wird einer mit der größten Tiefe ausgewählt. Bild 11 verdeutlicht das am Beispiel des 8-Zahlen-Puzzles. Damit sich das Verfahren nicht in zu große Tiefen *verrennt*, werden Schranken für den größten Tiefenwert gesetzt. Das Tiefenverfahren läßt sich mit dem Backtrakking-Verfahren vergleichen. Der Unterschied besteht darin, daß bei der Graphensuche immer eine Menge von Nachfolgeknoten generiert wird, während es beim Backtracking nur ein solcher Knoten ist. Wenn die Tiefengrenze erreicht ist, werden alternative Knoten derselben oder geringerer Tiefe weiter verfolgt.

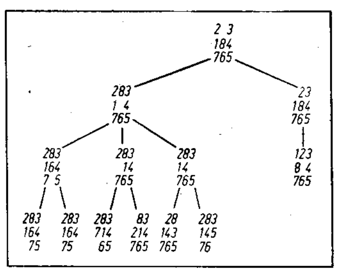


Bild 11 Tiefensuche

Breitensuche
Bei diesem Verfahren werden die Knoten mit der geringsten Tiefe zuerst weiterverfolgt. Dieses Verfahren findet garantiert den kürzesten Lösungsweg, wenn ein solcher existiert. Bei möglichem parallelen Verfolgen der Expansionen der einzelnen Nachfolgeknoten ist dann auch mit der geringsten Rechenzeit zu rechnen. In Bild 12 wird das Verfahren am 8-Zahlen-Puzzle demonstriert.

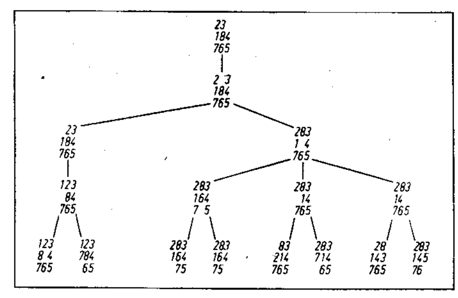


Bild 12 Breitensuche

Heuristische Suchverfahren

Bei der Breiten- und Tiefensuche sind in der Regel viele Knoten zu durchmustern, bis die Lösung erreicht ist. Das sprengt in den meisten Fällen die Grenzen, die durch Zeit und Speicherraum vorgegeben sind. Daher sind sie in Systemen der Künstlichen Intelligenz nur selten anwendbar.

Unter Nutzung heuristischer problemabhängiger Informationen ist es mitunter möglich, den Aufwand gewaltig zu reduzieren.
Dieser Aufwand für einen Lösungsweg ist charakterisiert durch die Kosten der einzelnen Bögen zwischen den Knoten, die sich auf diesem Weg befinden: c(k(i),k(j)). Es gilt, einen minimalen Weg zwischen a und e(i) zu finden – und das mit minimalem Suchaufwand. Die Kombination dieser beiden Kriterien bestimmt die heuristische Mächtigkeit eines Suchalgorithmus.

Bewertungsfunktionen
Bewertungsfunktionen sollen bestimmen, welcher Knoten einer Menge vielversprechend zur Lösung führen könnte. Zur Demonstration dienen Brettspiele und Puzzles, in denen bestimmte Konfigurationen bestimmte Punktzahlen erhalten, je nachdem wie vielversprechend sie sind.
Es sei f(k) so eine Bewertungsfunktion für den Knoten k.
Für unser 8-Zahlen-Puzzle könnte man f(k) wie folgt definieren:
f(k)=t(k)+p(k) mit
t(k)-Tiefe des Knotens k und
p(k)-Anzahl der Ziffern, die nicht an ihrem Platz sind.
In Bild 13 ist ein Beispielbaum gegeben. In der Anfangskonfiguration

123
8 6
754

befinden sich 3 Zahlen nicht an ihrem Platz, folglich ist f(a)=3.

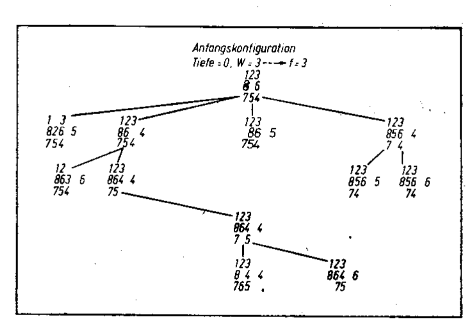


Bild 13 Suche mittels Berechnungsfunktion

*Der Algorithmus A**
Zuerst beschreiben wir den Algorithmus A. Unter dieser Bezeichnung ist er in die Literatur zur Künstlichen Intelligenz eingegangen. Wir definieren die Funktion f(k) als Kosten des minimalen Weges vom Anfangsknoten a durch den Knoten k zu einem Endknoten e(i).
Wir definieren uns weiterhin folgende Hilfsfunktionen:

m(k,e(i)) – Kosten des minimalen Weges von k zu einem bestimmten Endknoten;

$$h^*(k)=\min_{\{e(i)\}}[m(k,e(i))];$$

$$g^*(k)=m(a,k).$$

Dann ergibt die Funktion $f^*(k)=g^*(k)+h^*(k)$ die Kosten eines optimalen Weges vom Anfangsknoten zu einem Endknoten über einen Knoten k. Unsere Funktion f(k) sei dann gegeben durch f(k)=g(k)+h(k), wobei g(k) die Summe der Kosten der einzelnen Kanten von a zu k ist (für den kleinsten Weg, der bisher gefunden wurde) und somit gilt $g(k)>g^*(k)$, und h(k) trage eine gewisse heuristische Information über das Problem, ähnlich wie p(k) in der Bewertungsfunktion. Der Algorithmus A nutzt diese Funktion f(k) als Bewertungsfunktion.
Für den Fall, daß h(k) eine untere Schranke von $h^*(k)$ ist, sprechen wir vom Algorithmus A*. Dieser Algorithmus findet einen optimalen Weg zu einem Endknoten, wenn dieser existiert. Ein Spezialfall dafür ist die h-Funktion, die identisch Null ist. Dann liegt Breitensuche vor, und dieses Suchverfahren findet ja auch garantiert den kürzesten Lösungsweg.
Mit Hilfe heuristischer Suchverfahren, der Algorithmen A und A*, lassen sich auch kompliziertere Probleme mit akzeptablem Aufwand lösen. Dabei ist nicht gesagt, daß A* in jedem Fall vorteilhaft ist, beispielsweise erhalten wir ja für h=0 die Breitensuche, und hier wird eine erhebliche Menge Knoten mehr entwikkelt als eigentlich nötig. Es muß deshalb darum gehen, bei der Suche die heuristische Komponente der Berechnungsfunktion so einzugrenzen, daß möglichst wenige Abweichungen vom Lösungspfad nach links und nach rechts auftreten.
Zum Vergleich können Sie am folgenden Beispiel die Wirkung der heuristischen Komponente untersuchen. Wir haben vom 8-Zahlen-Puzzle folgende Anfangskonfiguration gegeben:

281
463
75

Als Berechnungsfunktion, das heißt, als deren heuristische Komponenten h(k), sollen folgende dienen und verglichen werden
a) h=0 (Breitensuche)
b) h(k)=p(k) (siehe unter Bewertungsfunktionen)
c) h(k)=s(k)+2r(k), wobei s(k) die Summe der Entfernungen (nötige Verschiebungen) von einer Zahl in der Konfiguration zu ihrem Platz in der Endkonfiguration ist. In der Anfangskonfiguration ist s(a)=1+2+2+1+1+0+2+1=10 (die einzelnen Summanden entsprechen den Zahlen in Uhrzeigerrichtung).
r(k) ist eine Summe, deren Summanden entsprechend ihren Zahlen in der Konfiguration gebildet werden, und zwar:
1 für den Fall, daß die Zahl vor ihrem Nachfolger steht, also 1 vor 2, 2 vor 3, ..., 8 vor 1;
2 für den Fall, daß die Zahl im Mittelfeld steht;
3 für den Fall, daß die Zahl nicht vor ihrem Nachfolger steht.
Damit ist r(a)=3+1+3+3+3+3+3+2=21 und h(a)=52.
Alle drei Formen der Funktion h führen mit einer Tiefe 12 zum Ziel. Während eine ungenaue Abschätzung für a) einen Graphen mit über 1000 Knoten ergeben würde, sind es bei b) bereits nur 65 Knoten und c) liefert nur 24 Knoten, also einen spürbar geringeren Aufwand.

Suchen in UND/ODER-Graphen
Grundbegriffe

Das Suchen nach einem Lösungspfad in UND/ODER-Graphen ist im Wesentlichen für zerlegbare Produktionssysteme bedeutsam. Ein v-Verbinder führt von einem Elternknoten zu einer Menge von v Nachfolgeknoten (UND-Knoten), die durch das Aufsplittern der Elterndatenbasis entstanden sind. Von einem Knoten können mehrere solcher v-Verbinder (mit unterschiedlichen v-Werten) ausgehen und bilden damit eine Menge von ODER-Nachfolgeknoten. Bild 14 zeigt einen solchen UND/ODER-Graphen. Die Begriffe

UND- bzw. ODER-Knoten sind jedoch nicht mehr ganz exakt; ein UND-Knoten kann durch entsprechende andere Verbinder auch ein ODER-Knoten sein und umgekehrt. Beispielsweise ist k(8) ein UND-Knoten bezogen auf k(5) und k(3) und ein ODER-Knoten bezogen auf k(4). Es ist einzusehen, daß es dann auch mehrere Lösungsgraphen geben kann. Angenommen, {k(9),k(10)} erfüllt die Zielbedingung. Die Bilder 15a und 15b zeigen zwei verschiedene Lösungsvarianten. Beide unterscheiden sich durch den Kostenaufwand, der für die Lösung nötig ist. Dieser kann rekursiv ermittelt werden. Für Abbildung 15b erhalten wir

$\mathbf{c=c(k(1))+c(k(4),K)+c(k(5),K)}$, wobei c(k(1)) den Kosten des 2-Verbinders entspricht, der von k(1) ausgeht, und c(k(4),K) den Kosten von k(4) bis zur Menge der Zielknoten K. So ist

$$\mathbf{c=2+c(k(4),K)+c(k(5),K)}$$
$$\mathbf{=2+1+c(k(6),K)+2+c(k(8),K)+c(k(9),K)}$$
$$\mathbf{=5+2+c(k(10),K)+c(k(9),K)+1+c(k(9),K)+c(k(9),K)=8.}$$

Hierbei wurde als Kostenwert für einen v-Verbinder der Wert v angenommen.

Heuristische Suche
*mit dem Algorithmus AO**

Die heuristische Suche in UND/ODER-Graphen erfolgt ähnlich wie in einem normalen Graphen. Von einem Anfangsknoten a ausgehend werden durch alle möglichen Verbinder die Nachfolgeknoten ermittelt. Von diesen muß eine Menge von Nachfolgeknoten ausgewählt werden (eine Menge von v Knoten, die durch einen v-Verbinder entstanden sind). Dazu sind gewisse heuristische Informationen nötig. Es sei h*(k) der Kostenaufwand für einen optimalen Lösungsgraphen vom Knoten k zu einer Menge von Zielknoten. h(k) sei ein Abbild dieser Funktion, und h(k) sei für jeden Knoten mit einem bestimmten Wert gegeben (heuristische Komponente).

Wir ermitteln von einem Knoten k die Nachfolgeknoten, mit denen sich Kosten verbinden, die dem h-Wert entsprechen. Die Kosten für k erhalten wir entprechend der genannten Formel $\mathbf{c_i(k)=c(v)+c(k(1))+\ldots+c(k(v))}$, wobei c(v) die Kosten der entsprechenden Verbinder sind und c(k(j)) entweder gleich h(k(j)) ist oder bereits über diese Formel berechnet wurden. Das wird für alle von k ausgehenden Verbinder getan und c(k) dann auf das Minimum aller Werte $c_i(k)$ gesetzt. Der Verbinder, über den dieses Minimum erreicht wurde, wird gekennzeichnet. Sollte vorher ein anderer von k ausgehender Verbinder schon einmal gekennzeichnet worden sein, so wird diese Kennzeichnung wider gestrichen, da sie nicht dem Minimum entspricht. Wenn c(k) von einem vorher für diesen Knoten berechneten (über einen anderen Weg zu k) oder von dem durch h(k) bestimmten Wert abweicht, werden die Kostenwerte der Elternknoten bis hin zum Anfangsknoten korrigiert und gegebenenfalls die Kennzeichnungen der Verbinder verändert. Ist dieses Verfahren abgeschlossen, kann mit der weiteren Expansion von Nachfolgeknoten fortgesetzt werden, bis die Lösung erreicht wurde. Abbildung 16 zeigt die Anwendung von AO* auf den Graphen in Abbildung 14.


wird fortgesetzt

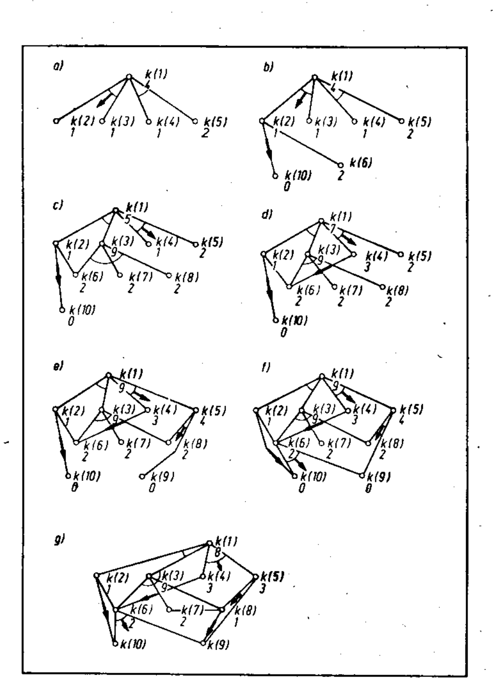


***Bild 16** Der AO*-Algorithmus*

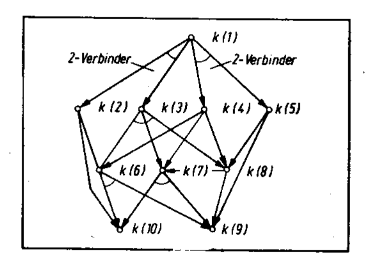


***Bild 14** UND/ODER-Graph*

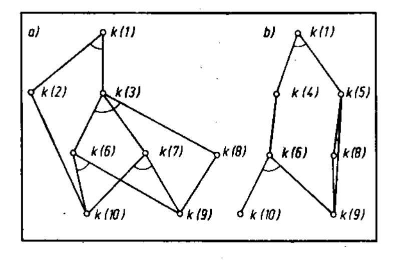


***Bild 15** Zwei Lösungen in einem UND/ODER-Graph*

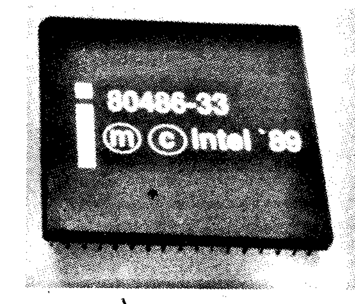
✉ KONTAKT ☏

Technische Universität Karl-Marx-Stadt, Sektion Informatik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 66 85 21

---

vorgestellt Der 80486 und seine Familie

oder „Was macht man mit einer Million Transistoren?"

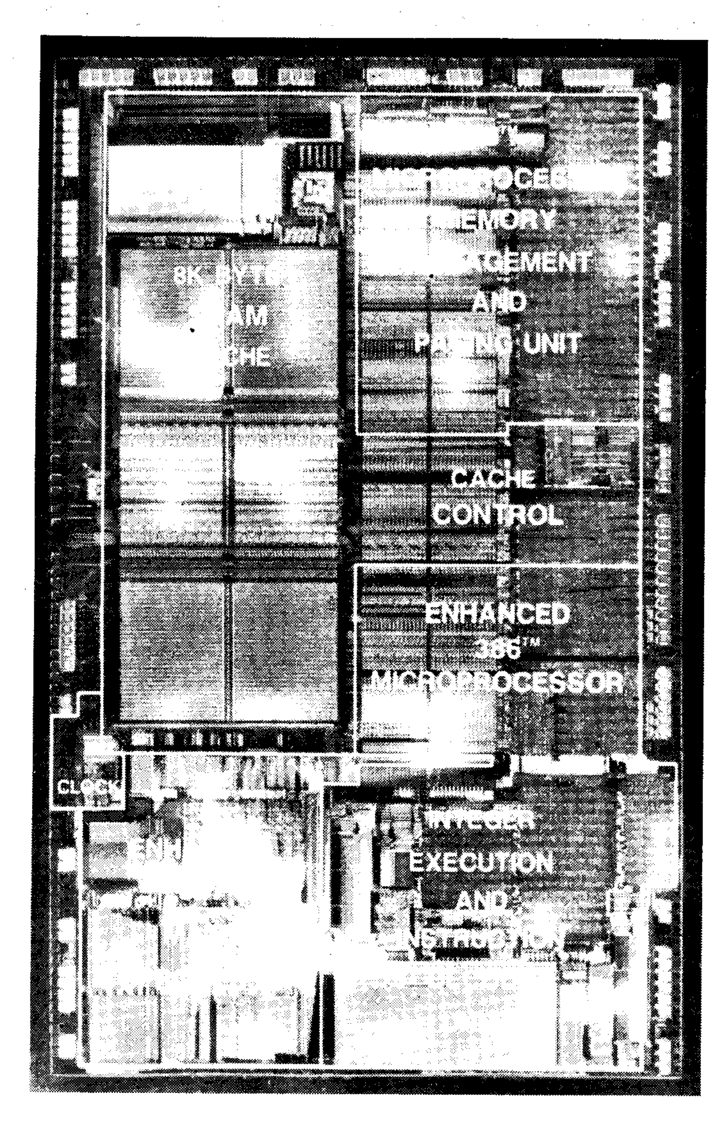


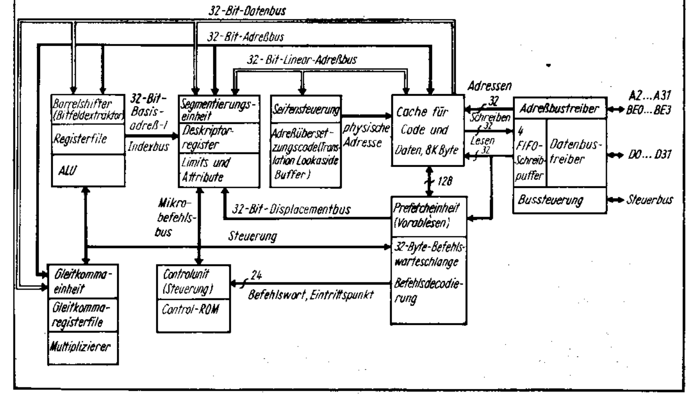
Mitte der 80er Jahre zeichnete sich die Möglichkeit ab, wenige Jahre später auf einem Logikschaltkreis 1 Million Transistoren zu integrieren, und ein Intel-Manager warf die Frage auf: „Was werden wir mit dieser Million Transistoren anstellen?"

Intel hat inzwischen das Technologieniveau 1 μm CHMOS-IV bei Logikschaltkreisen erreicht und gleich zwei Antworten auf diese Frage gefunden: Zum einen wurde mit dem 80860, dem „Supercomputer auf einem Chip" (s. MP 8/1989, S.251), eine neue Architekturlinie begründet, bei der mit den Mitteln der Höchstintegration drei Verarbeitungseinheiten als Multiprozessorsystem auf einem Chip zusammengefaßt werden konnten. Zum anderen war am 10. April dieses Jahres die Vorstellung des 80486 (nachfolgend 486 genannt). Er wurde mit einem Kostenaufwand von mehreren 100 Millionen US-$ in fünf Jahren entwickelt und setzt als Nachfolger des 386 die bewährte Architekturlinie 80×86 kontinuierlich fort. Das Programmiermodell wurde nur geringfügig erweitert; praktisch unterscheidet es sich kaum von dem des 386. Das mag auch als Zugeständnis an die Softwareentwickler zu werten sein, die noch immer vollauf damit beschäftigt sind, die mit der 386er Architektur gebotenen Möglichkeiten in den Programmen voll auszuschöpfen. Daß dennoch im Vergleich zum Vorgänger etwa viermal so viel Transistoren (genau: 1,2 Millionen auf 2 cm^2 Chipfläche) benötigt wurden, ist in der radikal weiterentwickelten internen Architektur begründet, woraus sich ein deutlicher Zuwachs an Verarbeitungsleistung ergibt: Bei gleichen Taktfrequenzen ist ein 486 etwa dreimal schneller als ein 386, und im Vergleich mit einem 8088 in einem IBM-PC bringt es der 486 sogar auf die 50fache Geschwindigkeit. Bei 25 MHz Taktfrequenz werden 15 bis 20 VAX-MIPS erreicht. Wie wurde der Leistungszuwachs gegenüber den Vorgängern erzielt?

1. Die Herstellungstechnologie des 486 (Intels 1-μm-Standard-CHMOS-IV-Prozeß) führt auf Grund der kleineren Strukturbreiten zu geringeren Gatterverzögerungen und zu reduzierten Signallaufzeiten, verglichen mit dem 1,5 μm-CHMOS-III-Prozeß der ersten 16-MHz-386-CPUs (inzwischen sind diese auch in der „schnelleren" Technologie als 33-MHz-Variante verfügbar). Der Prozessor kann deshalb von vornherein in einer schnelleren Version angeboten werden.

2. Die Ausführungseinheit des 386 wurde im 486 unter Anwendung von RISC-Techniken so modifiziert, daß einfache und relativ oft benötigte Transport- und Verarbeitungsbefehle ohne Mikroprogrammsteuerung direkt in Hardware in einem oder nur wenigen Taktzyklen ausgeführt werden. Komplexe Befehle arbeitet die CPU weiterhin intern mikroprogrammiert ab. Der 486 vereint so die RISC- und CISC-Architekturkonzepte ver-





einfachter und komplexer Befehlssatzarchitekturen miteinander.

3. Zur Parallelisierung der einzelnen Schritte der Befehlsabarbeitung wird das Fließbandprinzip (pipelining) eingesetzt: An 5 „Stationen" werden Befehle gelesen, zweistufig decodiert, ausgeführt und die Ergebnisse zurückgeschrieben. Eine gegenüber dem 386 doppelt so lange Befehlswarteschlange sorgt dafür, daß die Pipeline kontinuierlich mit Befehlen versorgt wird.

4. Um die hohe Verarbeitungsleistung auch bei der Verwendung verhältnismäßig langsamer, preiswerter DRAMs im Hauptspeicher ausschöpfen zu können, wurde ein 8 KByte großer Cache (Pufferspeicher) für Daten und Programmcode einschließlich Cachcontroller integriert. Damit können z. B. für ein Unterprogramm alle Befehle und Daten auf dem Chip gehalten werden, ohne daß Hauptspeicherzugriffe notwendig sind. Nur im Hintergrund der Programmausführung, daß heißt, ohne daß der Nutzer etwas davon bemerkt, wird dafür gesorgt, daß Ergebnisse automatisch in den Hauptspeicher zurückgeschrieben werden. Um den Cache möglichst schnell mit neuen Hauptspeicherinhalten füllen zu können, wurde ein Blocktransfer – der „Burstmode" – eingeführt, bei dem nur die Anfangsadresse eines einzulesenden Blocks an den Speicher ausgesendet und anschließend mit jedem Takt ein 32-Bit-Doppelwort übernommen wird (Busbandbreite 80 MByte je Sekunde bei 25 MHz). Die getrennte Adressierung von vier Speicherbänken gestattet eine überlappende Arbeitsweise bei Zugriffen auf aufeinanderfolgende Bytes.

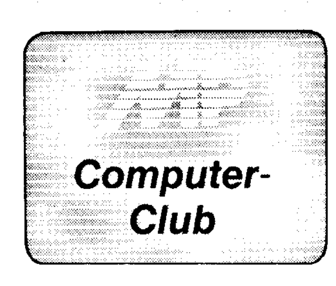
5. Auch rechenintensive Programme mit Gleitkommaverarbeitung werden wesentlich schneller ausgeführt. Der sonst übliche Arithmetikkoprozessor wurde als Gleitkommaverarbeitungseinheit (FPU) auf dem Chip gleich mit integriert. Die Kommunikation mit der CPU muß nun nicht mehr über Chipgrenzen geführt werden und gestaltet sich so einfacher. Zudem ist die FPU, ausgedrückt in Taktzyklen, durch weitere Architekturverbesserungen etwa vier- bis sechsmal so schnell wie ein 387. Sie realisiert jedoch nur eine Teilmenge des Funktionsumfangs des 387; für die Software hat das praktisch kaum Konsequenzen. CPU und FPU können selbstverständlich auch parallel arbeiten.

Darüber hinaus weist der 486 dieselben qualitativen Merkmale wie der 386 auf: 32-Bit-Datenbus (neu: mit Paritätsschutz), 32-Bit-Adreßbus mit 4 GByte physischem und 64 TByte (Terabyte) virtuellem Adreßraum, 32-Bit-Registerstruktur, gleiche Befehlssätze und Arbeitsmodi. Sechs neue Befehle sind vor allem für den Betriebssystemprogrammierer interessant. Drei Befehle beziehen sich auf die internen Caches. Ein Bytevertauschungsbefehl gestattet Datenzugriffe auf Speicher, in denen die Bytereihenfolge gegenüber den in der 86er Familie geltenden Konventionen vertauscht ist: Statt des „Little Endian"-Verfahrens (niederwertiges Byte vor höherwertigem Byte) kann so auch auf „Big Endian"-Formate zugegriffen werden (das höherwertige Byte zuerst, dann der Reihe nach die folgenden niederwertigeren Bytes). Man kann so den 486 z. B. in gemischten Mulitprozessorsystemen verwenden, in denen die Kommunikationspartner beide Repräsentationsformen nutzen. Auch die Verarbeitung von Zeichenketten (z. B. numerische ASCII- und BCD-Zeichenketten) vereinfacht sich. Schließlich gibt es noch zwei neue Befehle zur Ressourcenverriegelung in Form von Kombinationen aus Vergleichs- und Additions- sowie Austauschoperationen. Einige Steuerbits in den Controlregistern wurden neu definiert. Der 486 ist prädestiniert für den Einsatz in Hochleistungs-PCs, in Workstations und Servern. Auch in Mehrmikrorechnersystemen ist dieser neue Prozessor gut einsetzbar. Besonders die Betriebssystemgestaltung vereinfacht sich, selbst bei Kommunikation mit Prozessoren unterschiedlicher Typen. Ausgewählte Speicherbereiche (z. B. im Globalspeicher) können auch im höchstprivilegierten Modus gegen Schreiben und damit vor unberechtigten Datenänderungen durch Kommunikationspartner geschützt werden. Verklemmungsprobleme am Globalbus (zwei Prozessoren warten gegenseitig aufeinander) kann man durch Aktivieren eines Steuersignals lösen, durch das der Prozessor in den hochohmigen Zustand gesteuert wird und sich so vom Bus abkoppelt. Das nicht zu Ende abgearbeitete Programm kann später an der Unterbrechungsstelle fortgesetzt werden. Auch im Selbsttestmodus kann sich der 486 vom Bus abtrennen.

Und noch ein neues Detail weist der 486 gegenüber dem 386 auf:

Sowohl 16- als auch 8-Bit-Datenquellen sind auswertbar. Damit ist das gesamte Spektrum an Peripherieschaltkreisen der Vorgängergenerationen verwendbar. Für die 486er Familie wurde eine Reihe von Ergänzungsschaltkreisen angekündigt. Der 85C508 ist ein schneller Adreßbusdecoder mit 7,5 ns Gesamtverzögerungszeit. Als CSMA/CD-LAN-Controller (Ethernetprotokoll nach IEEE-802.3) kann der 82596CA mit integrierter Speicherverwaltung und einer zum Ethernet-Controller 82586 kompatiblen Speicherstruktur eingesetzt werden. Außerdem wird ein Chipsatz für die EISA-Schnittstelle angeboten. Dieser besteht aus folgenden Schaltkreisen: 82358-EISA-Bus-Controller (EBC), 82357 Integrated System Peripheral (ISP, dieser enthält einen DMA-Controller mit sieben Kanälen, eine Busarbitrationssteuerung, einen programmierbaren Interruptcontroller, einen Interruptgenerator für nichtmaskierbare Interrupts, die Refresherzeugung und -steuerung sowie fünf Zähler/Timer) und aus dem EISA-Bus-Pufferschaltkreis 82352 (EBB). Mit dem Bus-Master-Interfacecontroller 82355 (BMIC) wird der Aufbau von Zusatzkarten für den EISA-Bus vereinfacht. Der DMA-Controller 82480 hat acht unabhängige Kanäle und erreicht Transferraten bis zu 80 MByte/s bei 25 MHz. Ein neuer Cache-Controller soll künftig bis zu 512 KByte schnelle Caches für mehrere 80486 an 32- oder 64-Bit-Bussen verwalten. Der Preis des Prozessors soll bei 1500 DM liegen. Das ist weniger, als man sonst für ein „Pärchen" 386 + 387 bezahlen müßte. Die Taktfrequenz des 486 wird zunächst mit 25 MHz angegeben, 33- und 40-MHz-Versionen sind geplant. Ist nun der 486 „der Prozessor für den Rest des Jahrhunderts" (so Microsoft-Chef Bill Gates)? Sicher führt kein Weg an ihm vorbei. Für den OEM-Markt (einer Domäne von Intel) werden wieder vollständige Platinen (Motherboards) vorbereitet (das Entwicklerboard iSBC 486/125 DU für den Multibus II soll bereits erhältlich sein), neben Intel auch von Hewlett Packard, Olivetti, Compaq u. a. IBM hat die Einbindung des 486 in die PC/2-Architektur noch nicht bestätigt, das dürfte aber nur eine Frage der Zeit sein. Für die, denen der 486 noch immer zu langsam ist, stellt Intel noch eine Überraschung in Aussicht: In Zusammenarbeit mit Prime Computer soll 1992 eine ECL-Realisierung der 486er Architektur als Leiterkarte erhältlich sein. Mit Taktfrequenzen von mehr als 150 MHz werden 100 MIPS anvisiert. Dieser Leistungsbereich ist den traditionellen Großrechnern zuzuordnen. Auch Motorola plant ja für 1991 eine ECL-Version seines RISC-Prozessors 88000. Bezüglich der weltweiten und umfassenden Softwarebasis (Stichworte DOS, OS/2, UNIX V.5 usw.) hat Intel jedoch eindeutig die Nase vorn, so daß für die nächsten Jahre tatsächlich die Akzente bereits gesetzt sind. Übrigens: Die Planung für den 80586 ist bereits abgeschlossen. Das soll dann ein echter 64-Bit-Prozessor sein, und frühestens in 3 oder 4 Jahren wird man über ihn berichten können.

guv



Grafik mit Hardcopy

Als Ausgabegerät für grafische Darstellungen wird neben Plottern und Grafikbildschirmen der grafikfähige Hardcopy-Drucker nach wie vor gern benutzt. Das gilt für die 16-Bit-PCs ebenso wie für 8-Bit-PCs. Nachfolgend sollen Hardcopy-Routinen in Turbo-Pascal mitgeteilt werden, die als Prozeduren in einer Grafik-Toolbox die Ausgabe eines Grafikbildschirmes (GBS) sowie wahlweise eines das Grafikbild überlagernden Textbildschirmes (TBS) auf Druckern mit Standard-EPSON-Befehlscode und dem Typenraddrucker robotron SD 1152 realisieren.

Bei der Grafikarbeit unter Turbo-Pascal stehen leistungsfähige Tools zur Entwicklung von grafischen Darstellungen auf einem Grafikbildschirm zur Verfügung. Die Bildschirme sind im allgemeinen als Matrix – z. B. folgendermaßen – definiert:

```
CONST xmax=639;
      ymax=199;
      yzeile=24; (* ((ymax+1) DIV 8)-1 *)
TYPE
  Abb=ARRAY[0..yzeile,0..xmax] OF BYTE;
VAR GBS:Abb; (* Grafik-Bild-Schirm     *)
    TBS:ARRAY[1..24,1..80] OF CHAR;
```

Da es sich hierbei um RAM-Speicherbereiche handelt, kann man diese auch dynamisch mit GETMEM anlegen oder direkt auf den Bildwiederholspeicher – beispielsweise den TBS beim PC 1715 auf ABSOLUTE $F800 – legen.

Die Routinen zur Bedienung eines Grafikbildschirms, wie CLRGBS, CLRTBS, PLOT, LINE, QUAD, SCHRAFFIERE, KASTEN usw. sowie die Turtle-Grafik sollen hier als bekannt vorausgesetzt werden. Wir wollen davon ausgehen, daß im Grafikbildschirm GBS bereits ein Pixel-Bild enthalten sei. Wie dieses Bild dort erstellt werden kann, ist anderweitig in der Literatur bereits vielfach beschrieben worden. Wichtig ist nur der Grundgedanke, daß der Grafikbildschirm ein RAM-Speicherbereich ist, der nicht unbedingt mit dem visuellen Bildschirm übereinstimmen muß.

Durch Aufruf von

HARDCOPY (GBS, TBS, drpos, drmod);

wird der Inhalt von GBS und TBS auf den Drucker als Hardcopy ausgegeben. Setzt man im Aufruf z. B.

HARDCOPY (GBS^, null, drpos, drmod);

mit

CONST null:BYTE=0;

dann wird nur der Grafikbildschirm ausgegeben, der auch eine Pointervariable sein kann.

drpos: Linker Druckrand (z. B. Heftrand)

(Standardwert: drpos: CHAR=#0;)

drmod: Grafik-Modus (m=0..6 bei EPSON)

(Standardwert: drmod:CHAR=#4;)

Falls man für den ASCII-Zeichensatz beispielsweise den in /4/ vorgestellten 4×6-FONT benutzt, kann der Textbildschirm entfallen (null), da dann Text mit dem folgendem Prozeduraufruf in einer Auflösung von 4×6-Pixeln direkt in den Grafikbildschirm eingeschrieben werden kann.

DRAWSTRING (X1,Y1,<text>, <zahl>);

Der Vorteil hierbei ist, daß sich beim Verändern des Grafik-Modus (drmod) im Ausdruck die Schrift entsprechend mit verändert und nun an jeder beliebigen Stelle im Grafikbildschirm plaziert werden kann.

Mit der Prozedur HARDCOPY2 wird ein Bild doppelter Höhe in vertikaler Richtung auf dem Nadeldrucker aus-

Bild 1 HARDCOPY.INC für Druckergrafik mittels EPSON LX-86

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
(* ---------------------------------------------------------------------- *)
CONST
        BildschirmZeilen =24;
        BildschirmSpalten=80;
        Breite           =80; (* Pica-Schrift: 80 Zeichen/8"-Zeile *)
                              (* Elite-Schrift:96 Zeichen/8"-Zeile *)
        xmax :INTEGER=639;
        ymax :INTEGER=199;
        drpos:CHAR=#0;        (* Linker Druckrand, z. B. Heftrand  *)
        drmod:CHAR=#4;        (* Grafikmodus   m=0..6              *)

(* CASE drmod of               Grafik-Modus EPSON LX-86
   #0: Einfache Dichte   -->  60 Punkte/Zoll,  480 Punkte/8"-Zeile
   #1: Doppelte Dichte   --> 120 Punkte/Zoll,  960 Punkte/8"-Zeile
   #2: Doppelte Dichte   --> 120 Punkte/Zoll,  960 Punkte/8"-Zeile
                                               (Hohe Geschwindigkeit)
   #3: 4-fache Dichte    --> 240 Punkte/Zoll, 1920 Punkte/8"-Zeile
   #4:   CRT             -->  80 Punkte/Zoll,  640 Punkte/8"-Zeile
   #5: 1:1-Grafik        -->  72 Punkte/Zoll,  576 Punkte/8"-zeile
                                         (1:1-Grafik ---> Runde Kreise)
   #6: 1,5-fache Dichte  -->  90 Punkte/Zoll,  720 Punkte/8"-Zeile *)

(* ---------------------------------------------------------------------- *)

PROCEDURE hardcopy (VAR GrafikBS,TextBS;  drpos,drmod:CHAR);
(* Aufruf:
   ~~~~~~     HARDCOPY (gbs,tbs, drpos,drmod);  oder:
              HARDCOPY (gbs,null,drpos,drmod);  ohne Text-Bildschirm
                                            mit: const null:byte=0; *)
CONST dichte:ARRAY[0..6] OF BYTE=(60,120,120,240,80,72,90);
TYPE  A1=ARRAY[0..MaxInt] OF BYTE;
      A2=ARRAY[1..BildschirmZeilen,1..BildschirmSpalten] OF CHAR;
VAR   i,j,k,l,xdru  : INTEGER;
      z             : BYTE;
      gbs           : A1 ABSOLUTE GrafikBS;
      tbs           : A2 ABSOLUTE   TextBS;
      Flag          : BYTE ABSOLUTE TextBS;
BEGIN
IF BildschirmSpalten+ORD(drpos)<=Breite THEN l:=BildschirmSpalten
                                        ELSE l:=Breite-ORD(drpos);
k:=((ymax+1) DIV 8)-1;
IF xmax>=ROUND(8.0*dichte[ORD(drmod)]*(1.0-ORD(drpos)/Breite)) THEN
   xdru:=TRUNC(xmax-ORD(drpos)/Breite*8.0*dichte[ORD(drmod)])
                                                     ELSE xdru:=xmax;
FOR i:=0 TO k DO
BEGIN
write(lst,#27'J'#24);(* Zeilenabstand:  24/216 (Sofort eine Zeile *)
write(lst,#27'<');    (* Unidirektionaler Druck (eine Zeile        *)
write(lst,#13#27'f0',drpos); (* Leerzeichen fuer Heftrand          *)
write(lst,#27'*',drmod,chr((xdru+1) MOD 256),chr((xdru+1) DIV 256));
FOR j:=0 TO xdru DO BEGIN z:= gbs[i*(xmax+1)+j];
write(lst,chr(z))   END;
write(lst,#13#27'f0',drpos); (* Leerzeichen fuer Heftrand          *)
IF (i<BildschirmZeilen) AND (Flag<>0) THEN
             FOR j:=1 TO l  DO write(lst,tbs[i+1,j])
END;
writeln(lst)
END;
(* ---------------------------------------------------------------------- *)

PROCEDURE hardcopy2 (VAR GrafikBS,TextBS;drpos,drmod:CHAR);
(* Aufruf:  Bild mit doppelter Hoehe analog echter Bildschirmgrafik
   ~~~~~~   HARDCOPY2 (gbs^,tbs, drpos,drmod);  oder:
            HARDCOPY2 (gbs^,null,drpos,drmod);  ohne Text-Bildschirm
                                          mit:  const null:byte=0;  *)

CONST doppelt: ARRAY[0..15] OF BYTE=
            (0,3,12,15,48,51,60,63,192,195,204,207,240,243,252,255);
      dichte:ARRAY[0..6] OF BYTE=(60,120,120,240,80,72,90);
TYPE  A1=ARRAY[0..MaxInt] OF BYTE;
      A2=ARRAY[1..BildschirmZeilen,1..BildschirmSpalten] OF CHAR;
VAR   i,j,k,l,xdru  : INTEGER;
      z             : BYTE;
      gbs           : A1 ABSOLUTE GrafikBS;
      tbs           : A2 ABSOLUTE   TextBS;
      Flag          : BYTE ABSOLUTE TextBS;
BEGIN
IF BildschirmSpalten+ORD(drpos)<=Breite THEN l:=BildschirmSpalten
                                        ELSE l:=Breite-ORD(drpos);
k:=((ymax+1) DIV 8)-1;
IF xmax>=ROUND(8.0*dichte[ORD(drmod)]*(1.0-ORD(drpos)/Breite)) THEN
   xdru:=TRUNC(xmax-ORD(drpos)/Breite*8.0*dichte[ORD(drmod)])
                                                     ELSE xdru:=xmax;
FOR i:=0 TO  k DO
BEGIN
write(lst,#27'J'#24);(* Zeilenabstand:  24/216 (Sofort eine Zeile *)
write(lst,#27'<');    (* Unidirektionaler Druck (eine Zeile)       *)
write(lst,#13#27'f0',drpos); (* Leerzeichen fuer Heftrand          *)
write(lst,#27'*',drmod,chr((xdru+1) MOD 256),chr((xdru+1) DIV 256));
FOR j:=0 TO xdru DO         BEGIN z:=gbs[i*(xmax+1)+j] SHR 4;
write(lst,chr(doppelt[z])) END;
write(lst,#27'J'#24);(* Zeilenabstand:  24/216 (Sofort eine Zeile *)
write(lst,#27'<');    (* Unidirektionaler Druck (eine Zeile)       *)
write(lst,#13#27'f0',drpos); (* Leerzeichen fuer Heftrand          *)
write(lst,#27'*',drmod,chr((xdru+1) MOD 256),chr((xdru+1) DIV 256));
FOR j:=0  TO xdru DO        BEGIN z:=gbs[i*(xmax+1)+j] AND $0F;
write(lst,chr(doppelt[z])) END;
write(lst,#13#27'f0',drpos); (* Leerzeichen fuer Heftrand          *)
IF (i<BildschirmZeilen) AND (Flag<>0) THEN
             FOR j:=1 TO l  DO write(lst,tbs[i+1,j])
END;
writeln(lst)
END;
(* ---------------------------------------------------------------------- *)
```

Bild 2 HARDCOPY.DSW für Druckergrafik mittels Typenraddrucker SD 1152

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PROCEDURE HardCopy (VAR GrafikBS,TextBS; drpos,drmod:CHAR);
            (*    Fuer SD 1152   Typenrad - D r u c k e r      *)
(* Aufruf:
   ~~~~~~    HARDCOPY (gbs^,tbs, drpos,drmod);  oder:
             HARDCOPY (gbs^,null,drpos,drmod);
                  Ohne Text-Bildschirm mit:
                  CONST null:byte=0;                           *)
CONST BitMap      : ARRAY[0..7] OF BYTE=(1,2,4,8,16,32,64,128);
      Pixel       : CHAR='.'; (* Darstellungs-Character        *)
      MSchr       : BYTE=3;(* Horiz. Mindestschrittweite       *)
      Aufloesung  : STRING[3]='';(* Aufloesung in x-Richtung
                    durch Anhaengen von Blanks an die Pixel -
                    --> wird aus drmod::=#0..#6 bestimmt.      *)

TYPE  A1=ARRAY[0..MaxInt] OF BYTE;
      A2=ARRAY[1..BildschirmZeilen,1..BildschirmSpalten] OF CHAR;
VAR   i,j,z,xdru,RRand : INTEGER;
      gbs              : A1 ABSOLUTE GrafikBS;
      tbs              : A2 ABSOLUTE   TextBS;
      Flag             : BYTE ABSOLUTE TextBS;
      TABPos           : STRING[2];
      BlankPos,P3      : STRING[3];
                         (* SD 1152 minimal 3/60"-Schritte *)
      Bpos             : BYTE;
BEGIN
  RRand:=6*Breite;                (* Max. Punktzahl pro Zeile *)
  IF drpos>#9 THEN
                 TABPos:=CHR((ORD(drpos) DIV 10)+48)+
                         CHR((ORD(drpos) MOD 10)+48)
                 ELSE TABPos:='0'+CHR((ORD(drpos)+48));
  CASE drmod OF (* Behandlung der Aufloesung in x-Richtung  *)
    #0    : Aufloesung:=CHR(32)+CHR(32)+CHR(32); (* Doppelte *)
    #1,#2: Aufloesung:=CHR(32)+CHR(32);(* Hoehe:#0,#1 und #5*)
    #4,#6: Aufloesung:=CHR(32);
    #3    : Aufloesung:=''; (* Ohne zusaetzliche Verzerrung  *)
  ELSE   Aufloesung:=''; (* in x-Richtung                    *)
  END;
  BlankPos:='00'+CHR(48+MSchr+LENGTH(Aufloesung));
  IF drmod=#5 THEN
    IF  (xmax+1)*(ORD(BlankPos[3])-48)+6*ORD(drpos) > RRand
    THEN xdru:=(RRand-6*ORD(drpos))  DIV (ORD(BlankPos[3])-48)
    ELSE xdru:=xmax
                 ELSE
    IF ((xmax+1) DIV 3)*(ORD(BlankPos[3])-48)+6*ORD(drpos)>RRand
    THEN xdru:=3*(RRand-6*ORD(drpos)) DIV (ORD(BlankPos[3])-48)
    ELSE xdru:=xmax;
(* Max. Breite  12 Zoll  --> 720 Punkte a 60 Zeichen/Zoll
   Gedruckt werden aus 720 Punkten nur 720/3=240 Punkte als
   "EINS aus DREI" oder max. 180 Punkte (drmod::=#5) als
   "VIER fuer EINS".                                           *)
  writeln(LST);
  FOR i:=0 TO ((ymax+1) DIV 8)-1 DO BEGIN   (*    yzeile       *)
    FOR Bpos:=7 DOWNTO 0 DO BEGIN
      (* Drucker ruecksetzen:                                  *)
      write(LST,#$7F#27#91#48#32#75);(*Pica,10 Zeichen/Zoll *)
      write(LST,#27#91#48+TABPos+#96);
```

```
        (* G r a f i k  - Modus:                                   *)
        write(LST,#27#91#54#32#76);      (* 24 Zeilen  / Zoll  *)
        write(LST,#27#91#54#32#75);      (* 60 Zeichen / Zoll  *)
        write(LST,#27#91#48#48#49#101); (* Neue Zeile: 1/24"  *)
        IF (drmod=#0) OR (drmod=#1) OR  (drmod=#5)  THEN
        write(LST,#27#91#48#48#49#101); (* Doppelte Hoehe     *)
        P3:=''; z:=0;
        FOR j:=0 TO xdru DO BEGIN
          IF GBS[i*(xmax+1)+j] AND BitMap[Bpos] > 0  THEN
            IF drmod=#5 THEN Write(LST,Pixel+#27#91+BlankPos+#97)
                        ELSE P3:=P3+Pixel
                                                       ELSE
            IF drmod=#5 THEN Write(LST,#27#91+BlankPos+#97)
                        ELSE  P3:=P3+CHR(32);
          IF drmod<>#5 THEN BEGIN
                    z:=z+1;
                    IF z=3 THEN BEGIN
                    (* "EINS aus DREI" - es wird aus drei nur
                    ein Pixel jeweils  am Anfang gedruckt      *)
                    IF (P3=CHR(32)+CHR(32)+CHR(32)) THEN
                    Write(LST,#27#91+BlankPos+#97)  ELSE
                    Write(LST,Pixel+#27#91+BlankPos+#97);
                    P3:=''; z:=0
                                END
                            END
        END;  (* FOR j:=0 TO xdru ... *)
        IF drmod<>#5 THEN
        CASE z OF
             1:  IF P3=Pixel THEN BEGIN
                   P3:=Pixel+CHR(32)+CHR(32); Write(LST,P3) END;
           2:  IF (P3=Pixel+CHR(32)) OR (P3=CHR(32)+Pixel) OR
                   (P3=Pixel+Pixel)    THEN  BEGIN
                   P3:=Pixel+CHR(32)+CHR(32); Write(LST,P3) END;
        ELSE;
        END    (* END CASE z of          *)
      END;(* FOR BPos:=7 downto ...    *)
      (* Drucker ruecksetzen:                               *)
      write(LST,#$7F#27#91#48#32#75);(* 10 Zeichen/Zoll - Pica*)
      write(LST,#27#91#48+TABPos+#96);
      IF (i<BildschirmZeilen) AND (Flag<>0) THEN BEGIN
      (* Write(LST,#27#91#49#32#75);  12 Zeichen/Zoll - Elite *)
            FOR j:=1 TO BildschirmSpalten
                 DO write(LST,TBS[i+1,j])        END
    END; (* FOR i:=0 to ... *)
    Write(LST,#$7F#27#91#48#32#76); (* 6 Zeilen / Zoll        *)
    writeln(LST,#27#91#48#32#75);    writeln(LST)
 END;
```

gegeben, das der echten Bildschirmgrafik (CGA-Auflösung: 640 × 200 Pixel) maßstäblich entspricht.

Während beim EPSON-Drucker von einer Papierbreite von 8 Zoll ausgegangen wird (Breite = 80 Zeichen), kann beim SD 1152 auf 12 Zoll (Breite = 120 Zeichen) oder mehr orientiert werden. Die Steuerung der Auflösung in x- und y-Richtung wird auch beim SD 1152 über den Parameter drmod vorgenommen.

Anhand praktischer Versuche können die Standardwerte in den HARDCOPY-Prozeduren den Bedürfnissen des Nutzers noch angepaßt werden.

Falls man statt auf das Gerät LST erst einmal auf ein Textfile ausgibt, kann man die so aufbereitete Hardcopy später in Texte einbinden oder mit dem TYPE-Kommando bei zugeschaltetem Druckerecho ausdrucken.

Literatur

/1/ Grafik ohne Bildschirm. CHIP-Sonderheft „Turbo-Pascal", Vogel-Verlag, Würzburg

/2/ Plate, J.: Grafik-Grundlagen. mc (1986) 2, S. 78

/3/ Plate, J.: Abbildung dreidimensionaler Objekte. mc (1986) 2, S. 36–40

/4/ Plate, J.: Hochauflösende Drucker-Grafik. mc (1986) 9, S. 62

/5/ Cramer, E.: Erweiterter Bresenham-Algorithmus. mc (1987) 4, S. 66

/6/ Schmidt, H.: Schneller Drawline-Algorithmus. mc (1987) 6, S. 104

/7/ Hanisch, Ch.: Druckergrafik für technische Anwendungen. MP 1 (1987) 11, S. 330

Thomas Bauer

Programm Rollen links

Diese kurze MC-Routine für U 880 rollt den Inhalt des Bildschirms um genau ein Pixel nach links.

Dabei wird eine Bildschirmfläche von 255 × 255 Pixeln bewegt. Das Programm ist in verschiebbarem MC geschrieben und somit auf jedem freien Speicherbereich lauffähig.

Die Speicherplatzbelegung dient nur als Vorschlag.

0028 CD 18 F0 FD E5 F5 21 FF
0030 9F 06 FF C5 E5 FD E1 FD
0038 7E E1 07 3F 3F 06 20 CB
0040 16 2B 10 FB C1 10 EC F1
004B FD E1 CD 1B F0 C9 00 00

Die Eingabe des Programms kann über Modify auf einen beliebigen Speicherbereich erfolgen. Eine andere Variante wäre das Einlesen von Basic aus (siehe Bild).

Es existieren folgende Optionen:

POKE<STARTADRESSE dez>+19,X
X = 55 für herausrollen (Standard)
X = 63 für rundumrollen
X = 0 für invers rundumrollen

POKE<**STARTADRESSE** dez.>+8,Y
Y = **ANFANGSZEILE ROLLFENSTER** +128
(Standard ganzer Bildschirm, nur ungerade Zahlen ≦ 29,
1 Rollzeile = 8 Pixel hoch)

POKE<STARTADRESSE dez>+10,Z
Z = **(ZEILENANZAHL IM ROLLFENSTER)** * 8
(Standard ganzer Bildschirm, nur gerade Zahlen <30)

ACHTUNG:
Zeilenanzahl immer ≦ (32-Anfangszeile Rollfenster)

Der Aufruf des Programms erfolgt von Basic aus:
CALL < **STARTADRESSE** dez > bzw.
CALL * < **STARTADRESSE** hex. >

Thorsten Noske

```
65210 RESTORE 65300:STARTADRESSE = 40 :WINDOW 0,31,0,39:CLS
65220 FOR K=40 TO 78
65230 READ A:POKE K,A
65240 NEXT K
65300 DATA 205,24,240,253,229,245,33,255
65310 DATA 159,6,255,197,229,253,225,253
65320 DATA 126,225,7,63,63,6,32,203
65330 DATA 22,43,16,251,193,16,236,241,253,225,205,27,240,201,0
65340 FOR K=0 TO 25:PRINT AT(K,K);"ROLLEN LINKS":NEXT K
65350 POKE STARTADRESSE + 19,63
65360 FOR K=0 TO 255:CALL STARTADRESSE:NEXT K
65370 POKE STARTADRESSE + 10,16
65380 FOR I=1 TO 39 STEP 2:POKE STARTADRESSE + 8,I+128
65390 FOR K=0 TO 255:CALL STARTADRESSE:NEXT K
65400 NEXT I
```

Starten von Programmen aus Turbo-Pascal heraus

Turbo-Pascal enthält ab Version 4.0 eine Prozedur **EXEC**, die zum Aufruf weiterer Programme genutzt werden kann. Danach geht die Steuerung an das rufende Turbo-Pascal-Programm zurück /1/. In der Version 3.0 ist eine solche Funktion nicht enthalten. Für bestimmte Anwendungsfälle (z. B. Einschränkung der Nutzung) kann es durchaus sinnvoll sein, die Kontrolle der aufgerufenen Programme durch ein eigenes Programm vorzunehmen.

Mit der hier vorgestellten Lösung ist es möglich, beliebige Programme, Kommandos oder Stapeldateien aus einem Turbo-Pascal-Programm heraus abzuarbeiten, das anschließend die Steuerung wieder übernimmt. Die Grundidee ist nach /2/ der Aufruf von **COMMAND.COM** mit der Option /C.

C<**COMMAND.COM**/C **DIR** *·*

Der (zusätzliche) Kommandointerpreter **COMMAND.COM** wird geladen, führt das Kommando **DIR** *·* aus und wird wieder verlassen. Aus dem Betriebssystem heraus ist dieser Aufruf unnötig, da **COMMAND.COM** bereits geladen ist. Aus einem Anwenderprogramm heraus wird er sinnvoll, da dann **COMMAND.COM** selbst die Interpretation der Befehle und deren Ausführung übernimmt. Der Nachteil ist die Verwendung von zusätzlichem Hauptspeicher (hier etwa 24 kByte).

Das abgebildete Programm **EXEDEMO.PAS** ist eine modifizierte Version des in /3/ dargestellten Verfahrens. Zuerst wird der notwendige Speicher bereitgestellt (**SETBLOCK**), danach das Kommando eingelesen und an **EXEC** als Parameter zu **COMMAND.COM** übergeben.

Zu beachten ist, daß das Programm im C-Modus compiliert wird und die Optionen *mInimum free dynamic memory* und *mAximum free dynamic memory* auf den Wert von *StackSize* eingestellt werden, um eine ordnungsgemäße Rückkehr zu ermöglichen.

Literatur

/1/ Hübner, J.: MS-DOS. Verlag Technik, Berlin 1988

/2/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. Franzis Verlag, München, 1987

/3/ Rzędkowski, R.: Wywołania z Turbo-Pascala. Mikroklan 14 (1988) 2, S. 14

Jörg Sellmann

```
program exedemo;
{$C-}
const StackSize = $100;
type  Register = record
                   AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : Integer;
                 end;
      namestring = string[80];

var  reg : register;
     asciiz,par : namestring;

procedure setblock;     {Speicherbereitstellung}
begin
  with reg do
  begin
     bx:=Sseg-Cseg+StackSize; es:=Cseg; ax:=$4A00; msdos(reg);
     if odd(flags) then
       begin writeln('SetBlock Fehler,  AX=',ax); halt; end;
   end;
end;

procedure exec(parameter : namestring);

type par_type = record
                   env_seg,com_off,com_seg,d1,d2,d3,d4 : integer;
                end;

const par_block : par_type = (env_seg:0;com_off:0;com_seg:0;
                              d1:0;d2:0;d3:0;d4:0);

begin
   asciiz:='C:\COMMAND.COM'+#0; parameter:='/C '+parameter+#13;
   with par_block do
   begin com_seg:=seg(parameter); com_off:=ofs(parameter); end;

with reg do
  begin
     ds:=seg(asciiz); dx:=ofs(asciiz)+1; es:=seg(par_block);
     bx:=ofs(par_block); ax:=$4B00; msdos(reg);
  end;
end;

function akt : integer;     {aktuelles Laufwerk ermitteln}
 begin
    reg.ax:=$1900;
    msdos(reg);
    akt:=lo(reg.ax);
 end;

function up(z:namestring):namestring;     {Umwandeln in Großbuchstaben}
var i : integer;
begin for i:=1 to length(z) do z[i]:=upcase(z[i]); up:=z end;

begin {main}
setblock;
clrscr;
 repeat
  writeln; writeln; gotoxy(1,25); write(chr(akt+$61),':'); read(par);
  gotoxy(1,25);
  if par<>'' then if up(par)='EXIT' then HALT else exec(par);
 until false;
end.
```

MULTICAD-Kompendium

In Zusammenarbeit mit dem Kombinat Leuna und dem CAD/CAM-Zentrum des Hochschulwesens für die chemische Industrie an der TH Leuna-Merseburg wurde ein Kompendium zur Anwendung des CAD-Systems MULTICAD erarbeitet. Es ist maschinenlesbar nachnutzbar.

Technische Hochschule „Carl Schorlemmer" Leuna-Merseburg, Direktorat für Forschung, Otto-Nuschke-Straße, Merseburg, 4200; Tel. 46 29 22

Dr. Reinemann

Kopplung A 7150 mit Druckern über IFSS

Um vorhandene Breitwagendrucktechnik mit IFSS-Schnittstelle einsetzen zu können, wurde für 16-Bit-Rechner ein Treiber entwickelt und im DCP installiert. Hardwaremäßig wurde das Druckerkabel rechnerseitig mit einem Canon-Stecker bestückt und am Rechner der IFSS-Ausgang belegt. Die Druckgeschwindigkeit entspricht der des PC 1715. Breitwagendrucker K 6314 und SD 1152 können nur an den A 7150 angeschlossen werden.

VEB Leipziger Arzneimittelwerk, ORZ, Elisabeth-Schumacher-Straße 54/56, Leipzig, 7050; Tel. 6 44 41

Reimann

PENCIL und S 3004

PENCIL ist ein Programm für die Erstellung von Texten auf einem Kleincomputer vom Typ KC 85/1 oder KC 87 und die Ausgabe auf eine elektronische Schreibmaschine des Typs Erika S 3004. Dabei wird weder der Druckermodul des KC noch die Interfacebox der Schreibmaschine benötigt. Der Hardwareaufwand besteht lediglich im Verbinden des seitlichen PIO-Ports des KC mit der entsprechenden Buchse der Schreibmaschine über ein dreipoliges Kabel mit entsprechenden Steckverbindern. Mit dem Programm können die bearbeiteten Texte auf Magnetband ausgelagert und auch wieder eingelesen werden. Der mögliche Textumfang hängt von der RAM-Kapazität ab. Nach Betätigen der Taste **LIST** kann der mittels Kursortasten hineingerollte Bildschirminhalt und nach Betätigen der Taste RUN der komplette Text ausgegeben werden.
S 3004 ist ein Treiber für die Ausgabe auf der elektronischen Schreibmaschine S 3004 von einem KC 85/1 oder KC 87 aus. Nach Einordnen in das Betriebssystem mit dem Kommando ASGN LIST:=S 3004 ist der Treiber aktiv. Ein Paralleldruck zur Bildschirmausgabe ist über die Betätigung der Tasten CONTR P möglich. Der Hardwareaufwand besteht im Verbinden des KC mit der Schreibmaschine über ein dreipoliges Kabel mit entsprechenden Steckverbindern.

Kreiskulturhaus Schwedt, Abteilung Technik, Ernst-Thälmann-Straße 46–48, Schwedt (Oder), 1330

Reichenbach

dBase-Programm COMMWORK

COMMWORK ist ein universell anwendbares Hilfsprogramm zur Bearbeitung von dBase-Kommandodateien. Folgende Funktionen werden realisiert:
– Zeilenorientiertes Editieren von Kommandodateien; die betreffenden Programmzeilen werden vom Nutzer ausgewählt und können sofort bearbeitet werden.
– Auflisten oder Ausdrucken von Kommandodateien oder von ausgewählten Programmteilen
– Mehrere Kommandodateien oder daraus ausgewählte Teilabschnitte können zu neuen Kommandodateien verknüpft werden.
– Nach Eingabe einer Zeichenreihe werden nur die Programmzeilen gelistet, die die betreffenden Zeichenkombination enthalten.
COMMWORK wurde als dBase-Kommandodatei angelegt. Somit kann es jederzeit aufgerufen werden, ohne daß das Datenbanksystem verlassen werden muß. Alle Funktionen sind menügesteuert. Darüber hinaus steht dem Nutzer ein Helpmenü zur Verfügung, das die gesamte Projektdokumentation beinhaltet. Das Programm wurde auf dem PC 1715 mit dem Epson LX-86 im Betriebssystem SCP erarbeitet. Ein Drucker ist nicht zwingend notwendig. Mit einigen Änderungen ist eine Anwendung des Programms auch in REDABAS möglich.

VEB Metallwerk Wasungen, Abt. ODV, Katzweg 22, Wasungen, 6104; Tel. 2 14

Nitsche

Kopierprogramm COWA

Speziell den Anwendern, die zwar in der Lage sind, ihre fachbezogenen Programme perfekt zu nutzen, aber darüber hinaus kaum über spezielle EDV-Kenntnisse verfügen, soll das entwickelte Programm **COWA** helfen. COWA informiert über die Nettokapazität der Diskette, den belegten Platz, die maximale Anzahl der Directory-Einträge, die genutzten Einträge, die Größe und die Namen der gespeicherten Dateien und die zugeordneten Nutzerbereiche, und zwar auf einen Blick, ohne irgend etwas (außer der Laufwerksauswahl) tun zu müssen. Die Auswahl der zu bearbeitenden Dateien erfolgt per Tastendruck (und ist auch wieder genauso rückgängig zu machen). Unmittelbar nach Auswahl jeder zur Bearbeitung heranzuziehenden Datei wird auf die Auswirkungen (hinsichtlich der Kapazitätsinanspruchnahme) hingewiesen, die die geplante Operation hat (Löschen, Kopieren). Beim Kopieren wird neben dem Inhalt der Quelle auch der der Zieldiskette angezeigt. Eine eindeutige Menüführung in deutscher Sprache und eine sofort verfügbare Anwendungsbeschreibung helfen bei der Bedienung. Das Kopierprogramm COWA ist für die Rechner PC 1715, A 5120 und A 7100 verfügbar.

VEB Autobahnbaukombinat Magdeburg, Kombinatsleitung, Bereich ODV, Abt. DVP, Agnetenstraße 14, Magdeburg, 3024; Tel. 59 52 34

Wagner

MS-DOS-Grafiktoolbox unter Turbo-Pascal

Ein wesentliches Kriterium für die Akzeptanz von Programmen, welche die Grafikschnittstellen des Computers nutzen, ist die Übertragbarkeit auf Computer mit unterschiedlichen Grafikadaptern. Die Grafiktoolbox ermöglicht die Programmierung der GSX-Grafikschnittstelle (Betriebssystemerweiterung) von MS-DOS-Computern für unterschiedliche Grafikgeräte (Bildschirm, Plotter, Drukker...). Die Toolbox unterstützt folgende Grafikadapter:
– Robotron (A 7150/EC 1834)
– Hercules
– CGA.
In MP 11/1987, Seite 325, wurde bereits eine Toolbox für den A 7100 unter SCP vorgestellt. Die mit dieser Toolbox geschriebenen Programme sind nach Austausch der Toolbox weitgehend kompatibel und somit auf MS-DOS übertragbar. Die Toolbox liegt als Pascal-Quelltext vor und wird über eine Includeanweisung eingebunden.

Technische Universität Dresden, Sektion Elektrotechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4632917/4633189

Vetter

TEXTPLOT

Zur schnellen und sauberen Erzeugung von Texten für Poster, Wandzeitungen, Lehrmaterial oder Polyluxfolien wurde für den Kleincomputer KC 87 in Verbindung mit dem Kleinplotter XY 4131 ein Basic-Programm zur komfortablen Texteingabe, -korrektur (siehe MP 4/1988, S. 116), -abspeicherung und -ausgabe entwikkelt. Von Menüs aus erfolgt die Auswahl der Funktionen. Es können verschiedene Formate (A4 im Längs- oder Querformat und Polyluxfolien) mit oder ohne Überschrift (Schriftgröße 15 % größer) in den Schriftgrößen 3–15 mm und 15–43 mm sowie den Schriftarten Senkrecht-, Kursiv-, Normal-, oder Fettschrift realisiert werden. Insgesamt stehen 2 Programmversionen (TEXTPLOT-Schriftgröße 3–15 mm und TEXTPLOT-Schriftgröße 15–43 mm) und ein Hilfsprogramm (Kurzdokumentation) zur Verfügung.

Technische Hochschule Ilmenau, Sektion Phyteb, PSF 327, Ilmenau, 6300; Tel. 7 47 66

Dr. Spieß/Hecht

Arbeitshilfe für Turbo-Pascal 4.0/5.0

Für Rechner unter dem Betriebssystem MS-DOS bietet sich die Programmiersprache Turbo-Pascal mit hohem Leistungsvermögen und ansprechender Benutzeroberfläche an. Zur Unterstützung der Anwenderprogrammierung wurde eine Arbeitshilfe in Form einer Kurzdokumentation geschaffen, die umfassende Informationen zur integrierten Entwicklungsumgebung, den Units, sowie eine Übersicht der Prozeduren und Funktionen in kompakter Form enthält. Unterschiede zwischen den Versionen 4.0 und 5.0 sind gekennzeichnet. Der Umfang der Dokumentation einschließlich Register beträgt etwa 120 Seiten. Zur Nutzung der Programme **PKARC** und **PKXARC** (Archivierung und Entarchivierung) unter dem Betriebssystem MS-DOS liegt eine deutschsprachige Dokumentation vor. Die Dokumentationen werden auf 5¼"-Disketten des Nachnutzers übergeben.

INTERFLUG, Fachbereich WTI, Abt. Prozeßautomatisierung, Flughafen, Berlin-Schönefeld, 1189; Tel. 6 72 52 40

Henke

Einchiprechnersteuerungen

Die erste Lösung unter der Bezeichnung **KS-8** ist als Einkartensteuerung für Anwendungen der Prozeßsteuerung mit maximal 32 E/A-Signalen ausgelegt. Sie besteht aus einem Peripherie-Modul im Format 215 mm × 170 mm, auf den der Steuerungskern als Reiterkarte mittels Steckverbinders aufgesetzt wird. Der Steuerungskern (Format 105 mm × 100 mm) besteht aus dem EMR, dem Programmspeicher und der Takterzeugung. Der Peripherie-Modul enthält 16 binäre Prozeßeingabesignale, davon 14 optokoppler getrennt mittels MB 111 (Spannungsebene beliebig) mit jeweils eigenem Potential und mit 2 DL 004 gepuffert sowie 16 Prozeßausgangssignale, davon 12 optokoppler getrennt mit MB 104 (Spannungsebene beliebig bei 0,5 W Leistung pro Kanal) mit jeweils eigenem Potential und mit 4 DL 004 gepuffert. Die Signalbelegung der E/A-Kanäle wird mittels LED angezeigt. Die Vorteile der Lösung bestehen insbesondere im sehr geringen Materialaufwand, dem problemlosen Aufbau und guten Service-Bedingungen (Nachnutzungsgebühr 300,– M).
Die zweite Lösung unter der Bezeichnung **ECS 88** ist ein Steuerungssystem für die Prozeßautomatisierung, bei der die Anzahl der verfügbaren E/A-Leitungen unter Verwendung von Latches erweitert wurde. Für diese Erweiterung wurden die Ports P1 und P0 des EMR verwendet. Die theoretische Anzahl der möglichen E/A-Leitungen beträgt 128 × 8 Kanäle. Das Steuerungssystem besteht aus einem Rechnerkern mit einem Programmspeicher von maximal 4 KByte sowie den E/A-Karten mit jeweils 8 Ein- oder 8 Ausgangsleitungen. Die E/A-Karten gewährleisten Potentialtrennung und Pegelanpassung. Das System wurde auf Leiterkarten im Format 170 mm × 95 mm aufgebaut.
Die Vorteile der Konfiguration bestehen in der freien Verfügbarkeit von maximal 128 × 8 E/A-Kanälen und der Ports P3 und P2 des Einchipmikrorechners durch den Anwender sowie im relativ geringen Hardwareaufwand im Vergleich zu Steuerungsvarianten auf der Basis des U 880 (Nachnutzungsgebühr 500,– M).

Ingenieurbetrieb für die Anwendung der Mikroelektronik Bezirk Cottbus im VEB Mikroelektronik „Robert Harnau" Großräschen, Karl-Liebknecht-Straße 1, Großräschen, 7805; Tel. 60 15 (BfN) oder 51 17 (IfAM).

Übelhauser

Editor für 8 × 8-Font

Die Verwendung eigener Zeichenfonts für die Druckergrafik ist ein effektives Mittel zur besseren Gestaltung der Datenausgabe. Die Verwendung von Fontdateien für die Speicherung der einzelnen Zeichensätze bietet sich an. Um diese bei 16-Bit-Rechnern weit verbreitete Technik auch für 8-Bit-Rechner zur Verfügung stellen zu können, wurde ein spezielles Entwurfspaket entwickelt. Es besteht aus einem Fonteditor sowie einigen Musterfonts. Weiter werden Turbo-Pascal-Unterprogramme für den Zugriff auf Fontdateien aus Anwenderprogrammen heraus bereitgestellt. Der Fonteditor ist voll interaktiv aufgebaut und besitzt folgende menügesteuerte Grundfunktionen:
- Schreiben und Lesen der Fontdateien
- Auswahl eines zu editierenden Zeichens
- interaktives Editieren eines Zeichens
- Kopieren von Zeichen innerhalb des Zeichensatzes.

Die Fontdatei selbst ist als Binärfile aufgebaut. Die Nutzung unter Turbo-Pascal erfolgt als untypisiertes File. Die bereitgestellten Unterprogramme unterstützen das Einlesen der Fontdatei sowie die Ausgabe der Zeichen in einen internen Grafikpuffer des Anwenderprogramms. Als Zeichenfonts stehen gegenwärtig der IBM-Standardzeichensatz, ein Zeichensatz mit griechischem Alphabet und mathematischen Sonderzeichen sowie ein Zeichensatz mit verschiedenen Markern zur Unterstützung von Kurvendarstellungen zur Verfügung.
Das Editorpaket ist auf 5,25"-Diskette für PC 1715 unter CP/A lieferbar. Es umfaßt den Editor als .COM-Datei, die Unterprogramme für die Arbeit mit den Fonts als Include-Datei, die Fontdateien und eine Dokumentation.

Institut für Wirkstofforschung, Abt. FT, AG EDV, Alfred-Kowalke-Straße 4, Berlin, 1136; Tel. 51 63 2 21 *Höhne*

Compiler-Compiler MIRA

Der Compiler-Compiler unterstützt den Bau von Übersetzern für Fachsprachen. Vom Anwender ist eine attributierte Grammatik zu erarbeiten, die den gewünschten Compiler beschreibt. Durch die Verarbeitung dieser Grammatik mit dem Softwarewerkzeug MIRA wird der Kern eines Compilers, in Form von Quelltext der Programmiersprache C, automatisch erzeugt. Dazu zählen der Syntaxanalysator und die semantischen Aktionen. Die restlichen Module müssen per Hand geschrieben werden, es ist aber auch möglich, universelle Module für die lexikalische Analyse, die Fehlerauswertung und das Hauptprogramm aus einer Standardbibliothek zu übernehmen. Die vielfältige und leichte Handhabung von MIRA ermöglicht es ebenso, Interpreter automatisch zu generieren. Kennzeichen der erzeugten Übersetzer sind: effizienter und kleiner Code, geringe Laufzeiten und sprachunabhängige Fehlermeldungen. Der Compiler-Compiler MIRA ist unter WEGA (anderen UNIX-kompatiblen Systemen auf Anfrage) lauffähig. Im Unterschied zum UNIX-Tool yacc verarbeitet MIRA eine attributierte LL(1)-Grammatik. Weiterhin sind die Fehlerdiagnostik und das Aufsetzverhalten hervorzuheben. Vom Entwickler wird MIRA für die Generierung der Mensch-Maschine-Schnittstelle zur Steuerung digitaler Vermittlungsanlagen eingesetzt.
Für die Nachnutzung werden angeboten: Compiler-Compiler MIRA, universelle Module für lexikalische Analyse, Fehlerauswertung und Hauptprogramm, Testbeispiele sowie Anwenderdokumentation.

VEB Funkwerk Köpenick, ZFTN, Abt. EVR7, Edisonstraße 63, Berlin, 1160, Tel. 54 60 64 57 oder 54 60 63 32 *Kießner*

AUTRA-M16-Programmpakete

Das Programmpaket **Stücklistenauswertung** dient der Rationalisierung im Bereich der Konstruktion, der Technologie und der Produktionsplanung. Es erfüllt folgende Funktionen:
- Erfassen und Ändern von Stammdaten (Material- und Werkstoffdatei, Bauteildatei, Stücklistendatei)
- Erfassen und Warten von Manuskriptstücklisten, die redundanzarm gespeichert werden
- Bearbeiten von Stücklisten durch Zusammenstellung der Daten der Manuskriptstückliste und der Stammdateien, durch Berechnung von Fertig- und Einsatzmassen, Oberflächen sowie Preisen; die Strukturauflösung ist möglich.
- Erarbeiten von Bauteilauswahllisten nach frei wählbaren Merkmalen; eine Zusammenfassung mehrerer Stücklisten ist möglich.
- Anfertigen von Materiallisten auf der Basis einer oder mehrerer Stücklisten
- Der Belegdruck für TAB 1, TAB 2 und TAB 3 wird als Standard angeboten.

Zum Programmpaket werden folgende Schnittstellen für Anschlußprojekte angeboten:
- Anfertigen der Manuskriptstückliste unter CADdy
- Ausgabe aller Daten (Stücklisten, Bauteillisten, Materiallisten) als sequentielle Datei für die Weiterverarbeitung
- Übergabe der Daten an das AUTRA-M16-Programmpaket Arbeitsplanstammkarten

Für Erstanwender können Stammdateien für ausgewählte Halbzeuge und Normteile bereitgestellt werden.
Das Programmpaket **Arbeitsplanstammkarten** ermöglicht eine bedeutende Rationalisierung beim Anfertigen, beim Aktualisieren und beim Auswerten technologischer Fertigungsunterlagen. Es erfüllt folgende Funktionen:
- Speichern und Aktualisieren von Zeitnormativen sowie dazugehörigen arbeitsgang- oder bauteilspezifischen Texten; Berücksichtigen linearer und nichtlinearer Zeitverläufe
- rationelles Erfassen, Erzeugen und Aktualisieren technologisch orientierter Datenbestände (z. B. Arbeitsplanstammkarten); möglich ist dabei die bauteil- oder baugruppenbezogene Arbeitsweise. Auf der Grundlage gespeicherter Zeitnormative können die Normzeiten von Arbeitsverrichtungen automatisch ermittelt werden.
- Aggregation von Fertigungszeiten nach anwenderspezifischen Kriterien
- Übernahme bauteilbeschreibender Daten aus den Ergebnissen des AUTRA-M16-Programmpaketes Stücklistenauswertung
- variable Datenausgabe: Druck von Fertigungsbelegen (APSK, LS u. a.) und Datenausgabe auf Magnetdatenträger
- Kontrolle des Fertigungsdurchlaufs auf der Basis von Rückmeldungen aus den Fertigungsbereichen; Übernahme technologisch orientierter Datenbestände in eine Fertigungsplandatei
- Zwischen- und Endabrechnungen von Fertigungsprozessen.

Beide Programmpakete laufen unter MS-DOS. Die Übernahme von Daten aus der 8-Bit-Vorgängerlösung ist möglich. Für Erstanwender können Normativdateien mit ausgewählten Arbeitsverrichtungen bereitgestellt werden.

VEB Zentralinstitut für ökonomischen Metalleinsatz im VEB Qualitäts- und Edelstahl-Kombinat, Abteilung Produktionsvorbereitung, Karl-Marx-Straße, PSF 44, Dresden, 8080; Tel. 58 74 42 *Rotsche/Dr. Rothe*

DISKEDIT

Das gezielte Verändern einzelner Bytes auf ener Diskette mittels Standardsoftware ist sehr mühselig und kompliziert. Dafür wurde von uns das Programm **DISKEDIT** entwickelt. Es ermöglicht die Anzeige und die Korrektur aller Datenbytes einer Diskette einschließlich Systemspuren und Verzeichnis. Das Programm ist in Turbo-Pascal geschrieben, die Lauffähigkeit wurde unter SCP und CP/A geprüft. Bedingung ist ein Monitor mit möglicher Inversdarstellung (z. B. PC 1715), das Diskettenformat spielt keine Rolle. Die Verarbeitung erfolgt dateiweise. Mittels der Cursortasten kann man ähnlich wie im Textprozessor vor- und rückwärts durch die Datei blättern. Es steht eine Suchroutine nach einer Zeichenkette oder einer Folge von Bytes zur Verfügung. Ebenso kann durch Eingabe einer Blocknummer zum gewünschten Block verzweigt werden. Im Editiermodus kann der Cursor im aktuellen Block frei zu jedem dargestellten Byte bewegt werden. Dabei kann jedes Byte überschrieben werden, wobei vorher festgelegt wird, ob die Eingabe als Dezimalzahl, Hexadezimalzahl oder ASCII-Zeichen erfolgt. Bei Eingabe einer Spurnummer statt eines Dateinamens wird die Diskettenspur in der Reihenfolge der logischen Sektoren als Datei aufgefaßt und kann ebenso von Anfang bis Ende bearbeitet werden. Mit diesem Programm ist es problemlos möglich, durch Rechnerabsturz zerstörte Datenbestände oder Verzeichniseinträge zu reparieren, in nicht richtig abgeschlossene REDABAS- oder Pascal-Dateien die richtige Satzanzahl zu schreiben oder z. B. Programmausschriften zur Nutzerführung in die deutsche Sprache zu übersetzen. Das Programm ist durch eindeutige Menüführung weitgehend selbsterklärend.
Zum Nachnutzungsumfang gehören das Programm und eine kurze Beschreibung.

LPG(P) „An der Triebisch" Herzogswalde, Nr. 50, Helbigsdorf, 8211; Tel. Mohorn 290 *Ramm*

Turbolader für KC 87

Für die KC 87-Reihe wurde ein Turbolader entwickelt, mit dem alle Basic- und Maschinenprogramme (auch mehrteilig) normal oder beschleunigt kopiert werden können. Einstellbare Parameter sind:
- Aufzeichnungsgeschwindigkeit (bis 5fach)
- blockweises oder blockfreies (spektrumähnliches) Kopieren jeweils mit einstellbarer Geschwindigkeit
- Vortonpause zwischen den Blökken
- Autostart von Basic-Programmen
- Programmschutz.

Turbokopierte Programme werden im OS-Modus wie normale Programme geladen, denn sie besitzen am Anfang eine spezielle Laderoutine von 3 Blöcken. Beim weiteren Kopieren von Turbo-Programmen bleiben die Parameter erhalten.
Für die Nachnutzung ist eine ausführliche Dokumentation erhältlich.

Computerklub der SPS „Heinrich Hertz", Frankfurter Allee 14a, Berlin, 1035 *Lüdtke*

Magnetbandgerät CM 5300.01 über V24 an PC

Die Kopplung erfolgte an die Rechner A 7100/7150, ist aber für alle PCs mit V.24-Schnittstelle installierbar.
Der PC kommuniziert mit einem rechnergestützten Controller, der seinerseits das Magnetbandgerät ansteuert. Die Hardware des Controllers besteht aus einer Rechnerplatte (CPU U 880, 4 KByte EPROM, 60 KByte DRAM, CTC U 857, SIO UA 856, 2 PIOs U 855, Systemtakt 2,5 MHz) und einer Interfaceplatte (I/O-Logik, Treiber). Der Controller besitzt ein Urlade-Programm auf EPROM. Sein Betriebssystem wird über den PC geladen. Er erfüllt folgende Aufgaben:
- Empfang und Ausführung vom PC gesendeter Kommandos
- Datenaustausch zwischen dem internen Puffer und dem PC
- Meldungen über die Bandposition an den PC
- Steuerung des Magnetbandgeräts
- Prüfung der Daten auf Parität, CRC, LRC (Lesen) bzw. Erzeugung der Parität, CRC, LRC (Schreiben).

PC-seitig wird für die Magnetbandarbeit Software unter Turbo-Pascal (ab Version 3) in Form von Prozeduren und Funktionen auf zwei Ebenen angeboten:
- elementare Bandoperationen, d. h. Realisierung der Befehle des Controller-Betriebssystems auf unterster Ebene
- dateiorientiertes (logisches) Lesen, Schreiben und Positionieren des Magnetbands; Bandanalyse sowie Verzeichnisausgabe.

AdW der DDR, Institut für Kosmosforschung, Bereich 8, Rudower Chaussee 5, Berlin, 1199; Tel: 6 74 50 57 *Schrödter*

Fehlersuche am EC 1055.M

Durch den Einsatz der erarbeiteten Lösung (NV 381/88/Q) wird die Fehlersuche bei Hauptspeicherfehlern am EC 1055.M mit BC rationalisiert.

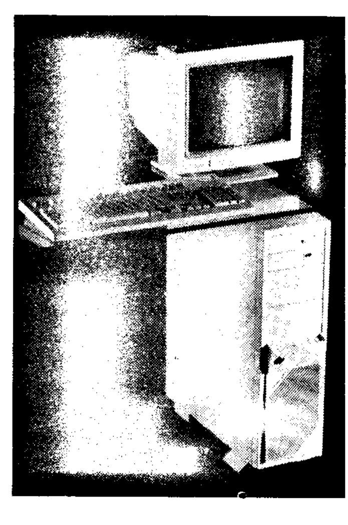
VEB Kontaktbauelemente und Spezialmaschinenbau Gornsdorf, Gornsdorf, 9163; Tel. 3 10 *Uhlmann*

Magabit-RAMs werden schneller

Mit der ständig steigenden Taktgeschwindigkeit von PCs entstand das Problem, daß der Prozessor (bei Taktraten von mehr als 10 MHz) auf den Hauptspeicher warten muß. Deshalb wurden für schnelle PCs Cache-Speicher eingeführt, die sich zwischen Hauptspeicher und Prozessor befinden. Diese Caches benötigen aber eine komplizierte Controller-Technik (siehe auch MP 10/1988, S. 311). Deshalb ist das Bestreben verständlich, die kostengünstigen dynamischen RAMs so schnell zu machen, daß der Cache ganz entfallen kann – vorausgesetzt, daß der RAM-Preis nicht wesentlich steigt.
Im Durchschnitt hatten dynamische RAMs bisher Zugriffszeiten zwischen 80 und 120 ns (der U 61000 vom VEB Carl Zeiss JENA bringt es auf 100 ns). Die Firma NMB Semiconductor kündigte im März dieses Jahres die Produktion eines 1-MBit-DRAMs mit 50 bis 70 ns für Anfang 1990 an. Im April meldete Hitachi die Entwicklung des 1-MBit-DRAMs HM571000JP mit einer Zugriffszeit von 35 ns, der noch Mitte dieses Jahres verfügbar sein sollte. Kurz darauf meldete sich IBM mit der Erfolgsmeldung: 22 Nanosekunden. Hierbei handelt es sich aber nur um Testchips, die vermutlich erst in einigen Jahren zur Serienreife gebracht werden können. Zur Verdrahtung dieses Chips werden zwei Schichten polykristallines Silizium und zwei Metallschichten verwendet; durch diese Kombination werden höhere Packungsdichten und damit höhere Geschwindigkeiten ermöglicht. Entwickelt wurde der Chip von IBM in Zusammenarbeit mit dem Tokyo Research Laboratory.
Aber auch bei den teureren statischen RAMs bleibt die Entwicklung nicht stehen. So konnte Toshiba in diesem Frühjahr die Entwicklung von Prototypen eines 1-MBit-Chips (in einer 0,8-μm-BICMOS-Technologie) mit einer Zugriffszeit von nur 8 ns bekanntgeben. Im März in Musterstückzahlen bereits verfügbar war der CMOS-SRAM HM624256 von Hitachi, dessen 256 K × 4 Bit eine Zugriffszeit von 35 ns aufweisen. *MP*

32-Bit-PCs von Schneider

In unserem Bericht von der Leipziger Frühjahrsmesse (MP 7/89, S. 221) hatten wir bereits erwähnt, daß die Schneider Rundfunkwerke ihre Produktpalette um ein Modell mit dem 32-Bit-Prozessor Intel 80386 erweitert haben; dieser ist in verschiedenen Versionen im Angebot.
Das Basismodell **80386-16/60** arbeitet mit einem 16-MHz-Prozessor und 0 Wartetakten, hat 2 MByte RAM (erweiterbar bis 16 MByte), ein 3,5-Zoll-Diskettenlaufwerk (1,44 MByte) und eine 60-MByte-Festplatte. Es besitzt eine serielle und eine parallele Schnittstelle, acht freie Steckplätze für Leiterkarten sowie eine Echtzeituhr.
Als Spitzenmodell ist der **80386-25/340** anzusehen. Der PC besitzt die 25-MHz-Variante des 80386, hat 4 MByte RAM (erweiterbar bis



24 MByte), ein 3,5-Zoll-Diskettenlaufwerk (1,44 MByte), ein 5,25-Zoll-Diskettenlaufwerk (1,2 MByte) und eine 333-MByte-Festplatte. An Schnittstellen bietet er zwei serielle, eine parallele sowie ein SCSI für bis zu sieben weitere Laufwerke.
Der Monitor der PC-Reihe hat eine Auflösung von 1024 × 768 Punkten und arbeitet mit einem Video-Adapter, der kompatibel zu CGA, Hercules, EGA und VGA ist. *MP*

Joint-venture zur Produktion von Halbleiterspeichern in den USA gegründet

Führende Elektronikunternehmen der USA haben vereinbart, gemeinsam die Produktion von DRAMs aufzunehmen. Nach Presseberichten wollen die Konzerne DEC, Hewlett-Packard, Intel, Natsemi, LSI Logic und AMD unter Führung von IBM eine gemeinsame Firma mit dem Namen **U.S. Memories Inc.** gründen, um die japanische Konkurrenz zurückzudrängen. In den nächsten Jahren sollen zur Produktion der DRAMs Investitionen in Höhe von mehreren Hundert Millionen Dollar vorgenommen und eine entsprechende Chip-Fabrik errichtet werden. Diese soll bereits Anfang 1991 die Produktion von 4-MBit-DRAMs unter Berücksichtigung der Erfahrungen IBMs aufnehmen. Kommentatoren in den USA verweisen darauf, daß mit der Bildung des Unternehmens eine Art Superkonzern entsteht, der den USA-Markt für derartige Chips nahezu vollständig kontrollieren könnte. Bisher hätten die miteinander konkurrierenden Firmen nur im Bereich der Forschung kooperiert.
In den vergangenen Jahren war es zwischen den USA und Japan wiederholt zu handelspolitischen Auseinandersetzungen im Bereich der Chip-Produktion gekommen. Dabei war von USA-Seite der marktbeherrschende Anteil japanischer Firmen an der Produktion von Hochleistungschips kritisiert worden, nachdem sich die US-Hersteller 1980 vom Speichermarkt allerdings selbst zurückgezogen hatten. W. Corrigan, Chairman von LSI Logic und Vorsitzender der Vereinigung, meinte nun: „Mit U.S. Memories meldet sich Amerika wieder im DRAM-Markt zurück.“ *MP*

Hewlett-Packard übernimmt Apollo

Seit dem 22. Mai ist die Apollo Computer Inc. eine hundertprozentige Tochter der Hewlett-Packard Company. Mit dieser Verbindung dürfte Hewlett-Packard im Workstationmarkt jetzt eine Spitzenposition einnehmen. Laut Dataquest lag HP mit seinen Workstations 1988, gemessen am Umsatz, weltweit auf Platz drei (in Europa auf Platz eins); Apollo lag auf Platz vier (in Europa auf Platz drei). Apollo wird als neue Division in den bisherigen Unternehmensbereich HP Workstations eingegliedert. HP übernimmt vorerst unverändert das komplette Produktprogramm von Apollo, das von der Personal Workstation bis zur Supercomputing Workstation reicht. Später ist die Zusammenführung aller Produktlinien beider Unternehmen geplant, einschließlich der Anlehnung an das kommende Betriebssystem der OSF. Die Produktpalette von Apollo wird bei HP als „ideale Ergänzung“ angesehen. Gemeinsamkeiten gibt es bei der Prozessorbasis mit Motorola-Prozessoren, die künftig auch den neuen 68040 berücksichtigen wird. HP/Apollo-Kunden können dann auf über 2500 Programme zugreifen, die jetzt auf Apollo-Domain- und HP-UX-Workstations laufen. HP will außerdem versuchen, von den Erkenntnissen der Apollo-Architektur PRISM zu profitieren, die ebenso wie HPs Precision Architecture ein RISC-Konzept ist. *MP*

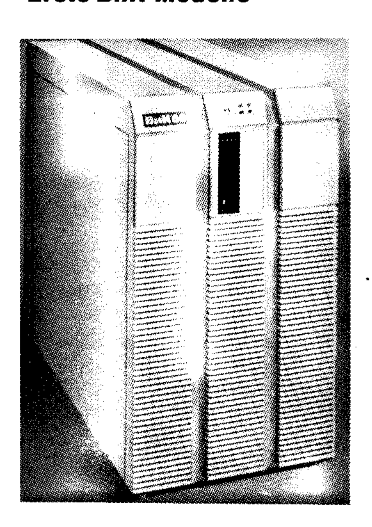
Personalcomputer im Aufwind

Die International Data Corp. (ICD) rechnet für dieses und das kommende Jahr mit einem ungebrochenen Wachstum beim Absatz von Personalcomputern in der Welt. Nach ihren Schätzungen wird der Verkauf von 20,922 Millionen PCs 1988 auf über 24 Millionen im Jahr 1990 steigen.
35 Prozent der 1988 ausgelieferten PCs arbeiteten mit dem Prozessor Intel 80286; der Anteil von 6 Prozent beim 80386er blieb hinter den Erwartungen zurück.
61 Prozent aller ausgelieferten Geräte waren IBM-PCs oder Kompatible; nur der Firma Apple sei es mit dem Macintosh gelungen, nennenswerte Marktanteile mit einem nicht kompatiblen PC zu erringen.
Der Anteil der Personalcomputer am weltweiten Umsatz von Datenverarbeitungsanlagen habe 1988 39 Prozent erreicht. *MP*

EPROMs bis 20 MHz

Bisher war es üblich, EPROM-residente Programme oder Daten, die häufig benötigt werden, in schnellere RAMs umzuspeichern, damit der Prozessor nicht auf den Speicher warten muß. Dem will Intel jetzt mit seiner neuen EPROM-Entwicklung begegnen. Die 256-KBit-Typen 27C202 und 27C203 in einer 1-μm-CHMOS-Technologie besitzen eine Bus-Schnittstelle mit Pipeline-Struktur. Der 27C203 für die 32-Bit-Prozessoren 80376, 80386 und 80386SX und der 27C202 für den 80960 KA/KB arbeiten mit diesen zusammen mit Taktfrequenzen von bis zu 20 MHz ohne Wartezyklen.
Ein Merkmal der 80386-Architektur ist das Vektorisieren (Pipelining) von Adressen. Der 27C203 arbeitet synchron mit dem 80386 und nutzt dessen überlappende Adressierung. Durch das parallele Ausgeben der Daten – während die Adressen eingelesen werden – wird die Busbandbreite erhöht. Die Zeit für den Übergang der 27C203-Pins in den hochohmigen Zustand (Tri-state-Zeit) beträgt nur 15 ns. *MP*

Erste BiiN-Modelle



Von dem im vergangenen Jahr gegründeten Siemens-Intel-Gemeinschaftsunternehmen BiiN (siehe auch MP 3/89, S. 66) wurden inzwischen die ersten Rechnersysteme vorgestellt: BiiN 20 und BiiN 60. Gemeinsames Merkmal ist eine neue Architektur, die sogenannten Mission-Critical-Anwendungen gerecht wird. Gemeint sind damit überdurchschnittliche Leistungen bezüglich der Fehlertoleranz, des Multiprocessing, der Objektorientierung, Echtzeit- und Transaktionsverarbeitung. Voraussetzung ist der spezielle Aufbau der Prozessorgrundmodule (die aus CPU, Speicher, E/A-Prozessor und dem schnellen Systembus bestehen), der eine enge Verbindung mit dem in ADA geschriebenen Betriebssystem BiiN/OS gewährleistet. Eine ganze Reihe wichtiger Betriebssystemfunktionen wurde dabei bereits hardwareimplementiert. Die 32-Bit-CPU ist eine Spezialentwicklung ähnlich dem 80960, im Gegensatz zu diesem jedoch nicht für andere Kunden verfügbar.
Das Modell BiiN 20 ist ein Desktoprechner mit ein bis zwei Prozessoren und bis zu 9 MIPS Leistung. Die Kapazität des Hauptspeichers beträgt bis zu 32 MByte, die der Massenspeicher bis zu 1,7 GByte, und der CPU-Adreßraum umfaßt bis 4 GByte.
Das Modell BiiN 60 (siehe Bild) kann mit zwei bis acht Prozessoren ausgestattet werden, hat eine Leistung von mehr als 40 MIPS, bis zu 80 MByte Hauptspeicher und eine Busleistung von 80 MByte/s. Die Plattenspeicher-

kapazität kann bis zu 40 GByte betragen. Bis zu 64 Terminals können direkt angeschlossen, und mit mehr als 1000 Terminals kann kommuniziert werden. *MP*

Molekulare Speicher aus organischen Molekülketten

Organische Moleküle bieten die Voraussetzung, um die Miniaturisierung von Bauelementen für die Informationsverarbeitung weiter vorantreiben zu können. Die Strukturen der Bauteile bewegen sich im Nanometerbereich. Die dreidimensionale Anordnung bewirkt eine weitere Verkleinerung um den Faktor 1000.
In verschiedenen Institutionen werden Moleküle hinsichtlich ihrer Eignung als Elektronikbausteine untersucht. Positive Ergebnisse wurden bei langgestreckten Molekülen erzielt, da bei diesen über eine Brücke aus Kohlenstoffverbindungen Elektronen von einem Ende zum anderen wandern können. Da diese Moleküle zwei verschiedene Zustände annehmen können, verfügen sie über die Voraussetzung für binäre Speicherelemente. An der TU München wird mit Photosynthese-Molekülen experimentiert, die in der Lage sind, einen gezielten Elektronentransfer über relativ große Distanzen vorzunehmen. Die Forscher selbst sind allerdings der Auffassung, daß die biologischen Moleküle zu kompliziert sind, um sie in der molekularen Elektronik einsetzen zu können.
Die Schwierigkeiten sollen nicht darin bestehen, geeignete Moleküle zu finden, sondern darin, die Ansteuerung zu realisieren. Durch Mikrolaser soll das Problem gelöst werden. Die Verbindung zwischen den molekularen Bauelementen soll durch sogenannte molekulare Drähte realisiert werden. An diesem Problem arbeitet das Max-Planck-Institut für Festkörperphysik. Die Forscher gehen davon aus, daß eine Elektronenwanderung in einer Kette aus Kohlenstoffatomen erfolgt, in der sich Einfach- und Doppelbindungen abwechseln. Treffen in dieser Kette zwei Einzelbindungen aufeinander, kommt es zu einer Störung der Elektronenverteilung, die sich wie eine Welle über die Molekülkette fortsetzt. Die Elektronenwanderung wird ausgelöst, wenn die Moleküle mit Licht angeregt werden. Trotz dieser Anfangserfolge ist die praktische Verwertbarkeit dieser Ergebnisse noch nicht in Sicht.

Quelle: Wirtschaftswoche vom 10. 3. 1989 *Wi*

Optische Rechner können elektronische Systeme nicht ersetzen

Während bisher die Meinung vertreten wurde, daß sich die optische Informationsverarbeitung nur abhängig von der elektronischen Informationsverarbeitung entwickeln kann, sind amerikanische Experten jetzt der Ansicht, daß es sich um zwei selbständige, unabhängige Bereiche handelt. Optische Computer können genausowenig elektronische ersetzen wie umgekehrt.
Die Experten schätzen ein, daß die Elektronik der Optik in ihrer Entwicklung weit voraus ist. Die Elektronik sei eine Technik, an deren Weiterentwicklung ständig gearbeitet wird und die auf „ausgereiften" Technologien und Komponenten beruhe.
Die optische Informationsverarbeitung beruhe auf völlig andersartigen Partikeln. Die Vorteile der Optik werden in
– der Freiraumverbindungstechnik
– der Verarbeitungsgeschwindigkeit und
– dem niedrigen Leistungsverbrauch gesehen. Sie sei eine dreidimensionale Technik, die ein besonders dichtes Packen der Komponenten ermögliche.
Nicht alle der genannten Vorteile lassen sich sofort realisieren. Ein Beispiel sei die Verarbeitungsgeschwindigkeit. Theoretisch liegt sie im Bereich zwischen 5 und 10 ps. Angestrebt wird aber der Femtosekundenbereich, der gegenwärtig nicht erreichbar ist.
Die optische Informationsverarbeitung sei der elektronischen auf den folgenden Gebieten überlegen:
– Mustererkennung
– neurale Netze
– Algebra
– digitale Verarbeitung
– Computer-Verbindungstechnik.
Während auf den Gebieten neurale Netze, Algebra, digitale Verarbeitung und Computer-Verbindungstechnik bereits Ergebnisse erzielt werden konnten, führten die Aufwendungen in der Mustererkennung noch zu keinem Erfolg. Das Problem Mustererkennung wird als derartig komplex bezeichnet, daß es mit den zur Verfügung stehenden Prozessoren nicht gelöst werden konnte.
Zwei Lösungswege werden als Möglichkeiten in Betracht gezogen: entweder ein einfacheres Problem finden (vgl. Fertigung) oder die optische Mustererkennungstechnik weiter verbessern, so daß sie, wenn auch entfernt, an die menschliche Fähigkeit heranreicht. Dies würde einen umfangreichen Komplex gespeicherter Muster voraussetzen sowie eine Selbstsortiermethode, so daß die wahrscheinlichsten Muster als erste verglichen werden könnten.
Die sehr leistungsfähige Datenbankverwaltung sei dabei ein Gebiet, auf dem die optische Verarbeitung Erfolg haben könnte.

Quelle: Blick durch die Wirtschaft vom 31. 3. 1989 *Fa*

Glasfaser der dritten Generation

Die optische Nachrichtenübertragung basiert gegenwärtig auf Laser-Lichtquellen, deren Herstellungskosten nach wie vor sehr hoch sind. Deshalb wird ständig an der Entwicklung neuer Übertragungsmedien gearbeitet.
Der Firma Philips ist es beispielsweise gelungen, eine neue Glasfaser mit der Bezeichnung *dispersion flattened single mode* (DFSM – streuungsreduzierter Einwellen-Lichtleiter oder abgeflachte Einmode-Faser) zu entwickeln. Dabei suchte die Entwicklergruppe nach einem Verfahren, mit dem sich die optische Spurbreite der Faser erweitern ließe, und nutzte dazu die PVCD-Technik. Mit diesem Verfahren wird das zur Herstellung des Lichtwellenleiters benötigte Glas mittels eine Plasmas aus der Dampfphase auskondensiert. Tausende ultrafeiner Glasschichten mit präzis vorhersehbaren optischen Eigenschaften lassen sich dadurch unmittelbar verschweißen, woraus vorteilhafte optische Eigenschaften resultieren. Die spektrale Bandbreite der DFSM-Faser beginnt wie die der Monomodems bei 1258 nm und reicht bis zu 1600 nm. Der große Vorteil der DFSM-Fasern besteht darin, daß sie wegen der höheren Toleranz kein streng monochromatisches Licht benötigen, sondern aus wesentlich preiswerteren LEDs gespeist werden können. Die höhere Bandbreite der DFSM-Fasern erlaubt außerdem die Ausnutzung einer zweiten Laseremission, die bei 1550 nm liegt. Bisherige Monomodems konnten nur standardmäßig bei 1300 nm die geforderte Qualität übertragen. DFSM-Fasern können demzufolge durch den zweiten Kanal doppelt so viel Informationen transportieren. Trotz der teuren Laserdioden erlangen Netze aus DFSM-Fasern eine höhere Wirtschaftlichkeit. Der Einsatz der DFSM-Fasern soll im Bereich der lokalen Telefonnetze erfolgen. Gegenwärtig arbeitet die Entwicklergruppe daran, die DFSM-Fasern auch für die Breitbandkommunikation über große Distanzen einzusetzen, wobei die hohen Dämpfungsverluste verringert werden müssen.

Quelle: highTech. – München (1989) 4 *Fa*

GaAs-Transistor als EEPROM-Element

Die Entwicklung eines GaAs-Transistors des Typs *Floating-Gate* soll den AT & T Bell Laboratories gelungen sein. Mit diesem Transistor könnte es erstmals möglich werden, nichtflüchtig arbeitende GaAs-Speicher aufzubauen. Darüber hinaus bietet er Eigenschaften, wie sie in EEPROMs zu finden sind. Das bedeutet, daß GaAs-Speicher mit dem neuen Transistorelement elektrisch umprogrammiert werden können. Außerdem verhalten sich Speicher dieser Art wie Festwertspeicher des ROM-Typs, die ihre Daten leistungslos halten. Die neuen Speicher erfordern allerdings, wenn sie in der EEPROM-Funktion arbeiten, eine Kühlung auf 140 K. Mit Leistungszuführung können sie zusätzlich auch als DRAMs oder SRAMs benutzt werden. Die Betriebstemperatur kann dabei bei 300 K (Raumtemperatur) liegen.

Quelle: Elektronik. – München 38 (1989) 3. – S. 7 *Fa*

Japan produziert 4-MBit-Speicher

Die japanischen Elektronikkonzerne beginnen gegenwärtig mit der Massenproduktion der 4-MBit-Speicher. In kurzer Zeit sollen beträchtliche Steigerungsraten erreicht werden, so daß mit den in der Tafel genannten Monatsproduktionszahlen gerechnet wird.

	9/89	12/89	1991
Hitachi Ltd.	200000	400000	
Toshiba Corp.		1000000	3000000
NEC	100000		

Die Firmen Mitsubishi Electric, Fujitsu, Oki Electric, Sanyo Electric und Sharp kündigten für Ende 1989 ebenfalls den Beginn der Massenproduktion der 4-MBit-Speicher an. Die Firma Matsushita Electronics will erst im April 1990 die Massenproduktion aufnehmen.

Quelle: Handelsblatt vom 18. 4. 1989 *Wi*

Dreidimensionale IC-Prototypen

An der Entwicklung dreidimensionaler integrierter Schaltkreise arbeiten u. a. die Firmen Mitsubishi, Matsushita und Sony, die bereits über Labormuster dieser ICs verfügen sollen. Die Firma Mitsubishi z. B. arbeitet gegenwärtig an der Erprobung eines ersten Image Sensor/Processor mit drei Lagen. In der obersten Lage befindet sich als Bildsensor eine Matrix aus Fotodetektoren, deren Anzahl zunächst auf 5 × 5 beschränkt wurde. In der Schicht darunter wurde eine AD-Umsetzerschaltung angebracht, und in der untersten Schicht befindet sich eine Logikschaltung, die arithmetische Funktionen übernimmt und mit deren Hilfe die Arbeitsgeschwindigkeit (Signal-Durchsatz) gesteigert werden kann.
Neben den Fertigungs- und Ausbeuteproblemen besteht das Hauptproblem bei der Entwicklung von 3D-ICs in der Wahl des Isoliermaterials, das sich zwischen den gestapelten Lagen der aktiven Elemente befindet. Es muß so beschaffen sein, daß auf ihm in weiteren Schritten monokristalline Strukturen ebenso wie in der untersten Lage erzeugt werden und gleichzeitig vertikale „Vias" als Signalverbindungen hergestellt werden können.

Quelle: Elektronik – München 38 (1989) 3. – S. 7 *Fa*

CrystalPrint Publisher

Speziell für den DTP-Markt entwickelte Qume jetzt den in MP 4/89 vorgestellten LCS-Drucker CrystalPrint weiter. Der CrystalPrint Publisher wurde dazu mit einem PostScript-kompatiblen Controller-Board ausgestattet, das erstmals den von Weitek für Grafikdrucker entwickelten HyperScript-Prozessor XL 8200 enthält. Der Drucker ist standardmäßig mit 3 MByte RAM, drei Schnittstellen und 35 residenten Schriften aus 11 Schriftfamilien ausgerüstet; die Auflösung beträgt 300 dpi. *MP*

dBase für DEC-Rechner

In Kooperation wollen der dBase-Entwickler Ashton-Tate und die Firma DEC eine dBase-Version für Desktop-Computer sowie für die komplette VAX-Familie entwickeln. Damit dürfte dBase erstmals für Multiuser-Computer und grafikorientierte 32-Bit-Workstations angeboten werden. *MP*

UNIX-Werkzeuge zur Programmentwicklung

von Christoph Polze, Reihe Technische Informatik, Berlin: VEB Verlag Technik 1989, 240 S., DDR 24,– M, Bestell-Nr. 5542786

In diesem Band der Reihe Technische Informatik werden in drei Schwerpunkten Aspekte des Betriebssystems UNIX und ausgewählte Werkzeuge behandelt. Im Mittelpunkt des ersten Teils (90 Seiten) steht die Beschäftigung mit der Dateiverwaltung von UNIX einschließlich der Beschreibung des Standardeditors ed und des Stream-Editors sed. Im zweiten Abschnitt folgt auf 45 Seiten eine Diskussion und Gegenüberstellung der beiden Kommandointerpreter Bourne-Shell und C-Shell. Die Darstellung dieser Komplexe ist von einer pragmatischen Herangehensweise gekennzeichnet, wobei sich die angegebenen Beispiele auf SCO-XENIX System V und WEGA 3.0 beziehen.

Im letzten Drittel des Buches wird die Werkzeugfamilie make, yacc und lex behandelt. Das breite Leistungsspektrum von make wird durch komplexe Skripten ausführlich demonstriert. Dieser Abschnitt ist allen UNIX-Anwendern sehr zu empfehlen, da die Einsatzmöglichkeiten von make sehr vielfältig sind. Der Zugang zu yacc und lex wird über die Beschäftigung mit einzelnen Beispielen vermittelt. Zwar werden nicht alle UNIX-Nutzer vor dem Problem der Generierung von Compilern stehen, aber gerade zu diesem Themenkreis schließt das Buch eine Lücke im Literaturangebot. Vielleicht wäre eine etwas ausführlichere Behandlung dieses anspruchsvollen Stoffes auf Kosten des größeren ersten Teils des Buches günstiger gewesen. Abgerundet wird dieses Kapitel durch die Angabe von Quelldateien für ein Mini-yacc im Anhang.

Sehr angenehm fallen der flüssige Stil und die sorgfältige Auswahl von deutschen Fachbegriffen auf. Die Vielzahl der angegebenen Beispiele fordert den Leser zu einer genauen Analyse heraus und regt zum Experimentieren an. Dem Verlag Technik ist zu danken, daß er dem immer größer werdenden Kreis der UNIX-Anwender mit diesem Band ein weiteres, sehr zu empfehlendes Arbeitsmittel in die Hand gibt. *U. Oefler*

Anwendungshilfen und Gestaltungsmuster für den optimalen Einsatz von Desktop Publishing

von G. Schnellhardt u. Ch. Dreyhaupt (Hrsg.), Interest-Verlag Kissing, 1989, Grundwerk etwa 400 S., inkl. Disketten 98,– DM

„Veröffentlichen vom Schreibtisch aus" – und das völlig ohne Schriftsetzer, Drucker und Verlag, so einfach stellt sich mancher Interessent „Desktop Publishing" vor. Kaum ein anderes Kürzel der gegenwärtigen EDV-Sprache wurde jedoch so oft fehlinterpretiert oder in einigen Veröffentlichungen der westlichen Fachpresse verzerrt dargestellt. Einige Meinungsäußerungen in nichttechnischen Zeitschriften haben zusätzlich Unklarheiten erzeugt.

Das Handbuch setzt genau an dieser Stelle an und gibt dem interessierten DTP-Anfänger Unterstützung und Hilfen sowohl bei der Auswahl des geeigneten Grundsystems als auch erste Anwendungshilfen bei der DTP-Gestaltung. Hier liegt die eindeutige Stärke dieses Nachschlagewerkes im Format eines A4-Ringbuchordners. Ein (erfahrener) Benutzer einer EDVA oder eines PC, der seine Anwendungen auf das Gebiet des DTP erweitern möchte, kann sich bestenfalls die technischen Voraussetzungen schaffen, er hat aber im allgemeinen kaum Kenntnisse über das umfangreiche und komplizierte Sachgebiet der Satz- und Drucktechnik, das neben dem Beherrschen der Fachtermini auch noch ästhetisches Gefühl und gestalterisches Können (Layout, Grafikgestaltungen) voraussetzt. Auch das vollkommenste DTP-Programm kann ihm solche Kenntnisse nicht abnehmen oder ersetzen.

Das Buch nun bietet dem Anwender Hilfestellung beim Erarbeiten von notwendigen Grundkenntnissen auf diesem Gebiet anhand didaktisch gut gewählter Gestaltungsbeispiele aus der Praxis (auf mitgelieferten Disketten, wahlweise 5,25 Zoll für MS-DOS-PCs oder 3,5 Zoll für den Apple Macintosh). Hier machen sich vorteilhaft die Erfahrungen der Autoren aus der langjährigen Praxis bei der Einführung von DTP bemerkbar – kaum ein anderes Werk der zahlreichen DTP-Literatur beschäftigt sich mit diesem für Neulinge so wichtigen Teilgebiet so ausführlich.

Auch zukünftigen Anwendern, die keine EDV-Spezialisten sind (beispielsweise Informations-/Dokumentationsstellen, kleinere spezialisierte Druckereien, Hersteller von Gebrauchsanweisungen, Dokumentationen, Produktbeschreibungen, Einladungen usw.), ist das Buch bei der Einsatzentscheidung eine gute Hilfe. In dem Nachschlagewerk werden die beiden gegenwärtigen Hauptlinien, nämlich DTP auf IBM-PCs (bzw. Kompatiblen) und DTP auf Apple Macintosh, jeweils getrennt nach Geräteausstattung und einigen wichtigen Programmen, vorgestellt und teilweise (auch mit Amortisationsberechnungen) miteinander verglichen. Wegen des Übersichtscharakters des Nachschlagewerkes verbietet sich ein tieferes Eindringen in die Standardsoftware Ventura Publisher oder Pagemaker. Zudem existiert hier bereits eine umfangreiche und gute Spezialliteratur. So beschränkt man sich vernünftigerweise darauf, die wichtigsten Stärken und Schwächen dieser Software, aber auch von zahlreichen Zusatzprogrammen, vorzustellen. Mit diesen Aussagen ist ein Anfänger des DTP in der Lage, mit einer Grundkonfiguration zu beginnen und spätere Erweiterungen oder Ergänzungen von vornherein mit einkalkulieren zu können.

Ein vorgesehener Erweiterungsservice, das heißt etwa vierteljährliche Erweiterungsbände zu jeweils aktuellen Themen (beispielsweise sind die Datenübertragung von Macintosh auf PC oder von PC zu PC, Farbscanner und Datenkompression geplant), wird aber dennoch dem vorgesehenen Anspruch, über möglichst alle Neuigkeiten der DTP-Technologie zu berichten, nicht gerecht werden können. Bereits die vorliegende Ausgabe muß Wünsche nach Vorstellung und Leistungsvergleich anderer bedeutender DTP-Programme offenlassen.

Das Fachwortlexikon mit Begriffen sowohl aus EDV als auch aus Satz- und Drucktechnik und Grafik erhöht den Wert des Buches für den Anfänger, auch wenn dort noch einige Ungereimtheiten oder nicht ganz zutreffende Definitionen enthalten und einige im Text genannte Begriffe nicht aufgeführt sind. Insgesamt kann das Buch jedem Kollektiv empfohlen werden, das sich kurzfristig in die umfangreiche Problematik des DTP einarbeiten muß. Auch Leitern, die in diesem Bereich Investitionsentscheidungen zu treffen haben, erleichtert das Werk eine schnelle und gründliche Einarbeitung. *Dr. H. Dolleschel*

Mit C zum Ziel: Das Buch für Ein- und Umsteiger

von A. Hickelsberger, Dr. Alfred Hüthig Verlag, Heidelberg, 1988, 192 S., 38,– DM, ISBN 3-7785-1555-1

Einige Ähnlichkeiten fallen dem von Turbo-Pascal verwöhnten Programmierer sicher auf, aber eben nur Ähnlichkeiten; nichts ist gleich oder gar identisch. Nicht nur, daß die Befehle anders heißen, sie wirken auch (zumindest teilweise) völlig anders.

Einer der wesentlichen Unterschiede zu anderen Programmiersprachen wird gleich zu Anfang deutlich: Jeder Ausdruck liefert einen Wert. Das führt zwar zum einen dazu, daß C-Programme recht kurz und knapp gehalten werden können, zum anderen beeinträchtigt dies die Lesbarkeit auch der eigenen Quelltexte erheblich. Umfangreiche Prozedurbibliotheken sind eine weitere Besonderheit der C-Compiler. Hierin sind viele wichtige, auf die spezielle Hardware des jeweiligen Rechners bezogene Standardroutinen enthalten. Damit sollen gute Vorraussetzungen gegeben sein, C-Programme portabel zu schreiben, was jedoch in der Praxis an der Vielzahl unterschiedlich leistungsfähiger C-Compiler scheitern kann, wenn die obere Leistungsgrenze ausgeschöpft werden soll. Der Programmierer ist dann oftmals gezwungen, Initialisierungen von Werten vorzunehmen, was sonst der Compiler übernommen hätte. Solche Grundoperationen aber, wie etwa Gleitkommaoperationen, sind in jedem Compiler möglich, zumindest softwaremäßig. Logische Konstanten jedoch sind nicht definiert; hier hilft man sich mit der Zuweisung von Werten zu Variablen, wobei ein Wert >0 für *wahr* und ein Wert $=0$ für *falsch* steht.

Dem Anspruch, für Umsteiger geschrieben zu sein, wird das vorliegende Buch durch sehr viele Bezüge zu anderen Programmiersprachen, insbesondere zu Pascal, aber auch zu Basic, gerecht. Der Programmierer wird dadurch immer wieder in gewohnte Umgebungen versetzt und vom Autor durch sehr viele Hinweise auf Fehlermöglichkeiten und Hilfen zu deren Vermeidung vor größeren System- und Nervenzusammenbrüchen bewahrt. Für den Einsteiger dürften meines Erachtens aber doch einige Probleme auftreten, denn das Buch nimmt sehr schnell an Niveau zu, und selbst bei kleinen Problemen müssen zahlreiche Komponenten, wie die unterschiedlichen Wirkungen der Funktionen und Operatoren und die Rang- und Reihenfolge ihrer Abarbeitung berücksichtigt werden.

Die Hinweise sowohl auf einen exakten Programmierstil als auch auf ein ausführliches Kommentieren der Programme sind insbesondere für die Sprache C dringend geboten.

C ist eine Sprache, die, wie wohl kaum eine andere, das Lehrbuch auf lange Zeit neben dem Computer liegen sehen möchte. In insgesamt 16 Kapiteln kommt man mit C vielleicht nicht gleich ganz zum Ziel, ihm doch aber ein deutliches Stück näher. Nach einer kurzen Einführung werden sofort die Sprachelemente, Variablentypen, Trenner, Operatoren und die Programmstruktur dargestellt. In einzelnen Abschnitten wird dann näher auf die Variablen und ihre Datentypen, die Operatoren, die Steuerstrukturen, die Speicherklassen, die Zeiger und Adressen und die Fehler, aber auch auf die Funktionen, die Strukturen und Varianten, die Bitfelder und Aufzählungen, den Preprozessor und auf die Dateibearbeitung eingegangen.

Neben den zahlreichen Beispielzeilen im erläuternden Text wurde ein gesonderter Abschnitt mit vier ausführlich kommentierten Beispielprogrammen zum Abtippen und Üben an sechster Stelle eingefügt, der in angenehmer Form eine Festigung des in den vorangegangenen Abschnitten Geschriebenen ermöglicht.

Zum Ende des fast 200seitigen Bandes werden noch je ein Abschnitt der Portabilität und dem ANSI-C-Standard gewidmet. Alles in allem ein Buch, das es verdient, dem Programmierer ein ständiger Begleiter am PC zu sein, damit er auch wirklich mit C zum Ziel kommt.

Der an der Programmiersprache C interessierte Leser sei darauf hingewiesen, daß die Titel „Programmieren in C" und „Unix und C" im Verlag Technik erschienen und in Bibliotheken verfügbar sind bzw. eventuell noch in Buchhandlungen erworben werden können. *J. Hill*



Bericht

Kolloquium zu sicherheitskritischer Software an der Verkehrshochschule

Die Hochschule für Verkehrswesen Dresden, WB Verkehrssicherungs- und Automatisierungsanlagentechnik der Sektion Prozeßautomatisierung, führte am 30. März 1989 gemeinsam mit der Technischen Universität Dresden, WB Systemsoftware des Informatikzentrums, dem VEB Werk für Signal- und Sicherungstechnik Berlin und dem Wissenschaftlich-Technischen Zentrum der Deutschen Reichsbahn ein Kolloquium zum Entwurf sicherheitssensibler Software nach der Methodologie *COSEM* (computergestützte, objektorientierte Systementwicklungsmethodologie) durch. Die bereits vor Jahrzehnten beklagte Softwarekrise ist Ausdruck der weitgehend unvorhersagbaren Kosten, von weit überschrittenen Terminvorgaben und nicht beherrschter Qualitätssicherung bei der Produktion und Nutzung von Softwareprodukten. Viele zu Methoden gewachsene Ideen wollten zwar einen Ausweg aus der Krise bieten, trugen aber nur zu einem im Vergleich zur Hardwareentwicklung sehr geringen Produktivitätszuwachs bei. Besondere Bedeutung erlangt die Frage der Softwarequalitätseigenschaften bei sicherheitskritischen Anwendungsfällen wie in der Medizin- oder Verkehrssicherungstechnik.

In den Jahren 1982/83 wurde am jetzigen Informatikzentrum der TU Dresden zur Lösung dieses Problems eine neue Herangehensweise ersonnen. Gemeinsam mit den obengenannten Institutionen und parallel zu ihrem Einsatz bei einem größeren Entwicklungsprojekt aus der Eisenbahnsicherungstechnik wurde sie weiterentwickelt. Das Kolloquium wolle den erreichten Stand dokumentieren.

Die „Meta-Methode" COSEM basiert auf dem Grundgedanken, den Entwurf durch Erstellen eines alle wichtigen Aspekte des Projektes umfassenden *Modells* zu vollziehen. Dabei werden zunächst die Sprachmittel und ihre semantischen Eigenschaften definiert (das *Rahmenmodell*), die dann zur – durch einen interdisziplinären Bearbeiterkreis durchschaubaren – Beschreibung der Entwurfsobjekte (das *Nutzermodell*) verwendet werden und umfangreiche Prüfungen am Modell zulassen. Das Nutzermodell erinnert an frameartig organisierte semantische Netze und erlaubt durch Anwendung prädikatenlogischer Funktionen die notwendige durchgängige Rechnerstützung der Teilschritte Konstruktion, Modifikation, Dokumentation, Prüfung und Verwaltung der einzelnen Entwurfsschritte. Dabei läßt die Variabilität des Rahmenmodells den Zuschnitt der einzusetzenden Methoden weitgehend offen. Der eigentliche Ziel-Code ist dann – idealerweise ohne menschliche Eingriffe – aus dem Modell heraus zu generieren. Durch diesen Ansatz können schon in der Konzeptionsphase und entwurfsbegleitend die gewünschten Qualitätsmerkmale (z. B. die *Korrektheit* als Teil der *Sicherheit*) erzwungen werden, anstatt sie im nachhinein in das Produkt „hineinzuprüfen". Dieser Modellgedanke wird inzwischen international stark diskutiert.

In seiner Einführung gab *Dr. Detering* (VEB WSSB) als „Stammvater" dieses Ansatzes einen Überblick zu Zielstellung und Entwicklungstendenzen sowie Anforderungen an Softwareentwicklungsumgebungen und ordnete COSEM hierin ein. Betont wurde die Nutzbarkeit der gleichen Mittel auch für die Umgebungsunterstützung (Technologiebeschreibung, Management). *Dr. Kühle* (TUD) beschrieb weitere theoretische Grundlagen und gegenwärtige Forschungsarbeiten am Informatikzentrum zum Ausbau von COSEM zur voll objektorientierten Meta-Methode COSEM+PLUS. *Wülfrath* (HfV) und *Henning* (WSSB) stellten eine Technologie für die Einbeziehung des Hardware-Entwurfes bzw. des System-Entwurfes in den frühen Phasen unter COSEM vor, die in ihrer Fortführung bis zum Aufsetzen auf eine Hardware-Beschreibungssprache getrieben werden soll (z. B. ASIC-Entwurf!). Die Arbeit mit funktionellen Modellen veranschaulichten *Müller* (WSSB) und *Dr. Lorenz* (HfV) durch das Beispiel eines Weichenantriebes aus der Eisenbahnsicherungstechnik. Die Festlegung von semantischen Merkmalen von Rahmenmodellen zur konstruktiven und analytischen Konsistenzsicherung von Modellbeständen war Gegenstand des Beitrages von *Franke* und *Dr. Protzner* (WSSB). *Ekkert* und *Dr. Zech* (WSSB) diskutierten die vorhandenen Werkzeuge zur Generierung eines zu COSEM-Modellen äquivalenten, ggf. instrumentierten Zielcodes. Die abschließenden Bemerkungen von *Müller*, Gruppenleiter Software im WSSB, faßten den gegenwärtig erreichten Stand zusammen und legten die bislang gesammelten Erkenntnisse bei der Entwicklung und Anwendung von COSEM dar.

Im Anschluß konnten sich Interessierte das Arbeiten mit Modellen, Konstruktions- und Prüfoperationen über Modellbeständen und die automatische Überprüfung in die Programmiersprache PLZ/SYS live an einem Arbeitsplatzcomputer A 7100 vorführen lassen.

Die lebhafte Diskussion zeugte von einem großen Interesse an der Softwaretechnologie. Zuhörer verwiesen auf die Aktualität dieses Ansatzes. Bemängelt wurde die noch nicht vollzogene Integration von Werkzeugen zur dynamischen Analyse wie Petrinetze, die vom Bearbeiterkollektiv bisher immer zurückgestellt werden mußte.

Dr. K.-A. Zech



Desktop Publishing und Druckerei

Mit der massenhaften Verbreitung von Personalcomputern lassen sich die bisher den Satzzentren der polygrafischen Industrie vorbehaltenen Arbeitsgänge der Erfassung und Gestaltung von Texten für gedruckte Publikationen auch dezentralisieren. Die modernen Desktop-Publishing-(DTP-)Systeme auf 16-Bit-Rechnern ermöglichen eine hohe Arbeitsproduktivität sowie eine den Ansprüchen vieler Publikationen genügende Gestaltung. Die Textdateien müssen nur einmal erfaßt werden, günstigstenfalls auf der Diskette des mit elektronischer Textverarbeitung arbeitenden Autors. Nach der redaktionellen Bearbeitung wird die Textdatei in das DTP-System eingelesen. Auf diesem läßt sich entsprechend den Möglichkeiten des DTP-Systems das fertige Dokument gestalten. Mit Hilfe von CAD-Software gezeichnete oder gescannte Grafik sowie Fotos können verwendet werden. Das Resultat ist (bei WYSIWYG-Anzeige) wirklichkeitsgetreu am Bildschirm zu verfolgen. Vom PC aus wird das Ergebnis in einer Auflösung von 300 dpi (dots per inch, Punkte je Zoll) auf einem Laserdrucker ausgegeben. Diese Auflösung genügt bereits vielen Ansprüchen. Soll eine höhere Qualität erreicht werden und ist eine größere Schriftenvielfalt gewünscht, muß die Ausgabe auf einem in Druckereien vorhandenen Satzbelichter erfolgen. Wie dieses möglich ist, wurde im März in einem unserer Druckereibetriebe vorgeführt. Mittels einer Probeinstallation stellten die Firmen GESYCOM, WISPI, und RANK XEROX ihre Technologie vor. So wurde die Online- und Offline-Kopplung eines LINOTYPE-CRT-300-Terminals mit einem DTP-System, bestehend aus einem AT-kompatiblen PC, dem Laserdrucker RANK XEROX 4045, der Software VENTURA-Publisher sowie Publikon und PCTronic gezeigt (im Bild ist die Kopplung eines CRTerminals 300 mit PC und Laserdrucker zu sehen). Fachleute aus Betrieben hatten die Möglichkeit, sich von der Leistungsfähigkeit des VENTURA-Publishers als Werksatzprogramm zu überzeugen. Auf PC 1715 erfaßte Textdateien wurden eingelesen und mit dem VENTURA gestaltet. Zur beabsichtigten Ausgabe auf einem LINOTYPE-Belichter wurde der Text vom VENTURA mit Hilfe der zuvor gespeicherten LINOTYPE-Dicktentabelle (Angaben zur spezifischen Laufweite der verwendeten Schriften) umgebrochen. Das geht auch bei deutscher Silbentrennung sehr schnell. Ein Probedruck auf dem Laserdrucker demonstrierte die Möglichkeit, die Drucksache in der fertig gestalteten Form zu begutachten und damit Material und Zeit bei der Satzausgabe zu sparen. Anschließend wurde die VENTURA-Datei mit Hilfe der GESYCOM-Software auf das Linotype-Satzsystem übertragen. Das ist online per Kabel oder offline mit Disketten möglich. Das Dokument enthält nach der Übertragung die LINOTYPE-Satzbefehle. Eine Nachbearbeitung auf dem Satzsystem ist möglich. Die Belichtung kann nun mit einer Auflösung von bis zu 1000 Linien pro Zentimeter auf Fotopapier oder Film erfolgen und entspricht damit auch den hohen Anforderungen der polygrafischen Industrie.

M. Fischer

Fachgruppe Forth im Kulturbund

Am 10. Juni 1989 fand in Leipzig die konstituierende Tagung der Fachgruppe Forth im Kulturbund der DDR, Computerklub Leipzig, statt. Unter Beteiligung interessierter Fachleute und Amateure aus allen Teilen der Republik wurde über Fortschritte in der Anwendung und Vervollkommnung eines Forth-83-Systems beraten, das von Mitgliedern der Fachgruppe für eine Reihe von Klein- und Heimcomputern implementiert wurde, darunter für alle KC-Typen, den AC 1 sowie den Z 1013. Hauptvorträge wurden zu den Themen Mulitasking, Gleitkommaarithmetik und Grafikanwendungen gehalten.

Die Fachgruppe Forth ist offen für jeden interessierten Computeramateur oder Fachmann auf dem Gebiet der Programmierung, der sich in seiner Freizeit mit der Programmiersprache Forth befassen möchte. Es finden monatliche Klubtreffen in Leipzig statt, auf denen Erfahrungsaustausch und gegenseitige Hilfe im Vordergrund stehen. Halbjährlich werden Tagungen zu speziellen Fragen der Forthprogrammierung veranstaltet. Daneben gibt die Fachgruppe für alle ihre Mitglieder schriftliches Informationsmaterial heraus.

Das oben erwähnte Fort-83-System kann von Interessenten gegen eine geringe Kopiergebühr bezogen werden. Kontaktadresse: M. Balig, PF 30–28, Großpösna, 7105.

Dr. M. Pfüller

INTERDECK '89

Erstmalig fand dieses Jahr in Karlsruhe in der Zeit vom 4. bis 6. April parallel zum Symposium der Anwenderorganisation DECUS München eine unabhängige Messe für VAX- und PDP-11-Anwender statt. Auf einer Fläche von 800 m² stellten über 95 Firmen ergänzende Hardware, kompatible Peripherie, Softwareprodukte und komplette Systemlösungen für Rechenanlagen der Fa. Digital Equipment Corporation (DEC) aus. Mit etwa 2200 Besuchern zählte die Messe nicht zu den größten; aber vom Veranstalter, der Network GmbH, wie auch von den Besuchern wurden die umfangreichen Möglichkeiten für Fachgespräche und Konsultationen hervorgehoben.

Schwerpunkte auf der Messe waren bei einer Vielzahl von Firmen vor allem Speichersubsysteme und ergänzende Kommunikationsprodukte. Eine führende Position auf diesem Sektor nimmt hier die Fa. Emulex (Bild 1) ein, die sich auf Controller jeglicher Art spezialisiert hat, die DEC-kompatible Interfaces und Protokolle emulieren und somit den Anschluß von Fremdperipherie an DEC-Rechner ermöglichen. Die Produktpalette umfaßt vor allem Massenspeicher und Kommunikationscontroller für UNIBUS- und Q-Bus-Systeme. Neu im Angebot sind SCSI-Adapter (Small Computer Systems Interface) zum Anschluß von SCSI-Peripherie an UNIBUS- und Q-Bus-Systeme.

Umfangreich ist auch das Angebot an Ethernet-Zubehör wie Transceiver, Repeater, Multiport-Repeater und Terminalserver. Ein interessantes Produkt stellte z. B. auch der Terminalserver Performance P 4000 (Bild 2) mit DEC-kompatiblem LAT-Protokoll (Local Area Terminal) zum Anschluß von 12 oder 16 Terminals mit einer max. Übertragungsrate von je 38,4 kBit/s an ein Ethernet-LAN dar. Mit einer Erweiterungsbox kann die Anschlußanzahl auf 32 Terminals bei gleichzeitiger Verringerung der Übertragungsrate auf 19,2 kBit/s je Terminal erhöht werden.

Zu den Spitzenprodukten zählt das SMDI-Subsystem (Storage Modul Disk Interconnect) zum Anschluß von schnellen SMD-Plattenlaufwerken der Fa. CDC an DEC-Rechner. Der SM-Umsetzer (z. B. SM70, Bild 3) setzt dabei das Standard Disk Interface (SDI) der DEC Storage Architecture (DSA) in das SMD-Interface um, wobei auf jedem SMD-Laufwerk ein RAxx-Laufwerk von DEC emuliert wird. In einem Schrank von ca. 1,5 m Höhe sind zur Zeit etwa 10,5 GByte Speicherkapazität realisierbar.

Auf der Messe war auch einer der bedeutendsten PC-Hersteller, die Firma Compaq, mit ihrem umfangreichen PC-Angebot auf der Basis der Schaltkreis-Familien 80286 und 80386 vertreten. Der Compaq Deskpro 386/25 (vorgestellt in MP 2/89) ist ein PC der oberen Leistungsgrenze mit einer Taktrate von 25 MHz bei voller 32-Bit-Architektur, max. 16 MByte Hauptspeicher und max. 1,2 GByte Plattenspeicherkapazität. Demonstriert wurde vor allem die Integration von PCs und PC-Netzen in Netze auf Basis von DECnet über die PCSA-Software (Personal Computer System Architecture).
Die transtec Computersystemvertriebs GmbH Tübingen hat eines der umfangreichsten Angebote an DEC- und DEC-kompatiblen Geräten für PDP-11- und VAX-Rechenanlagen. Neu im Angebot waren vor allem:

- Ergonomische Bildschirmterminals VDU 320 von Nokia Data (Bild 4)
- Hochleistungslaserdrucker von Hewlett Packard
- Thin Wire Multiport-Repeater für Ethernet von Cabletron
- 660-MByte-Winchesterplatten (5,25 Zoll) von CDC.

Das ergonomische Bildschirmterminal VDU 320 widerspiegelt die neuesten Entwicklungen für Bildschirmarbeitsplätze:
- flimmerfreies Bild durch eine relativ hohe Bildschirmwiederholrate von 72 Hz
- ermüdungsfreies Lesen durch die Darstellung von schwarzen Zeichen auf weißem Hintergrund
- bessere Lesbarkeit durch eine relativ große Bildschirmdiagonale von 15 Zoll (38 cm).

Mit einem interessanten und umfangreichen Angebot an speichersubsystemen und Netzwerkprodukten war auch die Fa. Micro Technology GmbH vertreten.
Der Q-Bus-kompatible Controller MQDX4 realisiert das MSCP-Protokoll (Mass Storage Control Protocol) zum Anschluß von Winchesterplatten- und Diskettenlaufwerken an MicroVAX- und MicroPDP-Rechnern. Er ist zum RQDX3 von DEC kompatibel. Bei voller Bestückung mit 4 Laufwerken (Bild 5) sind max. 3 GByte externe Speicherkapazität verfügbar. Weiterhin wurden auswechselbare Winchesterplattenspeicher MQD-19 zum Anschluß an Q-Bus-Systeme vorgestellt, wobei ein Laufwerk in der 5,25-Zoll-Technologie eine Kapazität von 660 MByte hat. Durch Herausnahme des hermetisch gekapselten Plattenstapels können die Daten analog einer Diskette geschützt werden.
Eine echte Alternative als Backup-Medium für die Datensicherung stellen die Kassettenmagnetbandlaufwerke auf der Grundlage der Video-Aufzeichnungsverfahren dar. Der Controller MTU-23 für UNIBUS-Systeme bzw. MTQ-23 für Q-Bus-Systeme (Bild 6) kann bis zu vier Kassettenmagnetbandlaufwerke EXB-8200 von EXABYTE bedienen. Auf einer kleinen 8-mm-Video-Kassette können etwa 2,3 GByte gespeichert werden. Pro Minute lassen sich etwa 7 bis 10 MByte von Platte sichern. MTU/MTQ-23 können von allen gängigen Betriebssystemen (RSX-11M, RSX-11M+, UNIX, ULTRIX, VMS) unterstützt werden.
Zahlreiche weitere Produkte werden gegenwärtig auf dem Netzwerksektor – wie Ethernetcontroller MEQNA, Ethernet-Adapter MESTA und Netzwerkanalysatoren LANguard – sowie auf dem Gebiet DSA-kompatibler Plattenspeicher- und Magnetbandsubsysteme angeboten.
Die SYNELEX Datensysteme GmbH war mit zahlreichen Netzwerkprodukten für LANs nach dem Ethernetverfahren vertreten. Insbesondere wurden DEC-kompatible Produkte von ISOLAN (Bild 7) angeboten wie
- Ethernet-Interconnect zum Anschluß von max. 8 Ethernet-Knoten über Transceiverkabel an ein Ethernet-Segment (kompatibel zu DELNI)
- entfernter Repeater zur Kopplung von zwei Ethernet-Segmenten über 1000 m Lichtwellenleiter (kompatibel zu DEREP)
- Multiportrepeater zum Anschluß von max. 8 Thin-Wire-Ethernet-Segmenten an ein (dickes) Ethernet-Segment (DEMPR).

Den technischen Möglichkeiten entsprach auch der LAN-Analysator LANalyzer EX 5600 von EXELAN auf der Basis eines portablen Compaq-PCs. Der LAN-Analysator kann alle gängigen Netzwerkprotokolle analysieren, Statistiken erfassen und verfügt über umfangreiche Diagnosehilfsmittel.

Die Fa. ROI (Rolf Obler Industrieelektronik) bot komplette Mikrorechnersysteme an, die PDP-11- und MicroVAX-kompatibel sind. Aus Eigenentwicklungen wurden Single Board Computer SBC J11-8 auf der Basis des DEC-Mikroprozessors J11 mit einer extrem hohen Packungsdichte und anderen Baugruppen vorgestellt (Bild 8).

STOLL EDV-Peripherie GmbH bot neben Terminals und verschiedenen Druckern vor allem hochleistungsfähige Laserdrucker F-1000, F-1200, F-2200 und F-3000 von Koycera an, die über eine Zusatzplatine voll kompatibel zum DEC LN03/PLUS sind. Der F-3000 hat eine Leistung von 18 A4-Seiten pro Minute und emuliert alle verbreiteten Druckertypen wie HP Laserjet, IBM Grafikdrucker, EPSON FX80, NEC Pinwriter u. a. Er verfügt über den Tektronix-4010/4014-Zeichensatz, Pixel-Grafik und 78 Festfonts in allen Emulationen sowie über 39 Barcodes.

Die Fa. EMC Computer Systems Ltd. stellte Speichersubsysteme aus wie etwa optische Plattenspeicher nach dem WORM-Prinzip (Write Once-Read-Multiple) von Archeion mit einer Kapazität von max. 56 GByte in einem Schrank. Die optischen Platten sind auswechselbar.

Eine weitere Neuheit auf der Messe war auch der erste löschbare und beliebig oft beschreibbare optische Plattenspeicher INSPIRE, der von der FORUM Technische Entwicklung und Vertrieb GmbH angeboten wurde. Ein Doppellaufwerk hat eine Kapazität von 1,3 GByte. Es werden optische 5,25-Zoll-Disks von der Fa. 3M verwendet, die auswechselbar sind. Die mittlere Zugriffszeit beträgt 83 ms und die max. Datenübertragungsrate 7M Bit/s. Das Interface gibt es für den Q-Bus und den UNIBUS. Es wird das MSCP-Protokoll realisiert.

Darüber hinaus wurden zahlreiche weitere interessante Produkte. vor allem Softwareprodukte und Systemlösungen ausgestellt, auf die hier im Rahmen dieses kurzen Berichtes nicht weiter eingegangen werden kann.

Prof. Dr. Th. Horn

***Bild 1** Emulex-Stand auf der INTERDECK '89; Massenspeichercontroller, Kommunikationscontroller und Speichersubsysteme für DEC-Rechner*

***Bild 2** Terminalserver Performance 4000 von Emulex mit DEC-kompatiblem LAT-Protokoll für 12 oder 16 Terminals mit max. 38,4 kBit/s, Ethernet-Anschluß IEEE 802.3-kompatibel*

***Bild 3** SMDI-Subsystem von Emulex mit SM70-Interfaceumsetzer zum Anschluß von CDC-Platten mit insgesamt max. 10,5 GByte, Emulation der DEC Storage Architecture*

***Bild 4** Ergonomisches Alpha-Terminal VDU 320 von transtec (Hersteller Nokia Data); 15 Zoll Diagonale, 72 Hz Bildwiederholrate, schwarze Zeichen auf weißem Hintergrund, entspiegelte Röhre, kompatibel zu DEC VT 100, VT 220 und VT 320*

***Bild 5** MQDX4 von Micro Technology; Q-Bus-kompatibler Controller mit MSCP-Protokoll zum Anschluß von 4 Winchesterlaufwerken mit insgesamt max. 3 GByte, volle Kompatibilität zu DEC RQDX3*

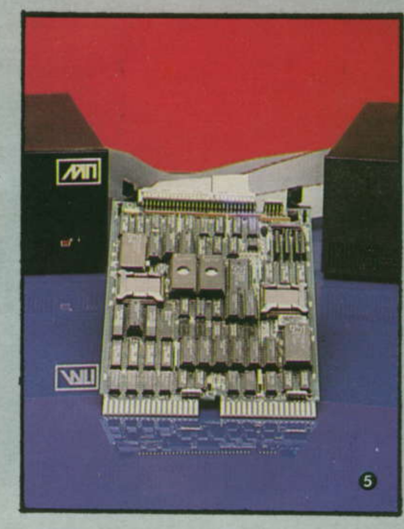
***Bild 6** MTU/MTQ-23 von Micro Technology mit Kassettenlaufwerk EXB-8200 von EXABYTE; Anschluß an DEC-Rechner mit Unibus oder Q-Bus. Max. Speicherkapazität 2,3 GByte pro Kassette, Transferrate 246 kByte/s*

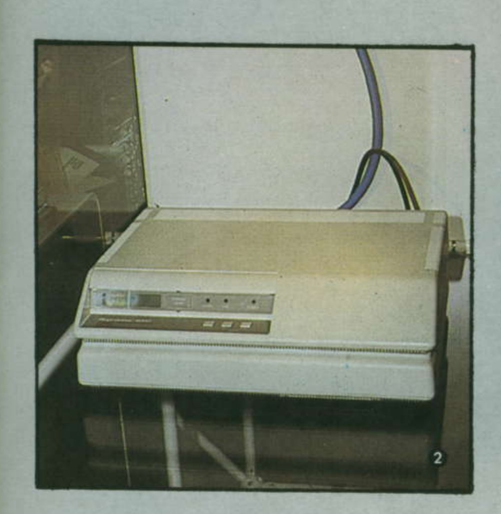
***Bild 7** Netzwerkprodukte von ISOLAN; Multiportrepeater für mehrere Thin-Wire-Ethernet-Anschlüsse, entfernter Repeater mit Lichtwellenleiter, Ethernet Interconnect für mehrere Stationen*

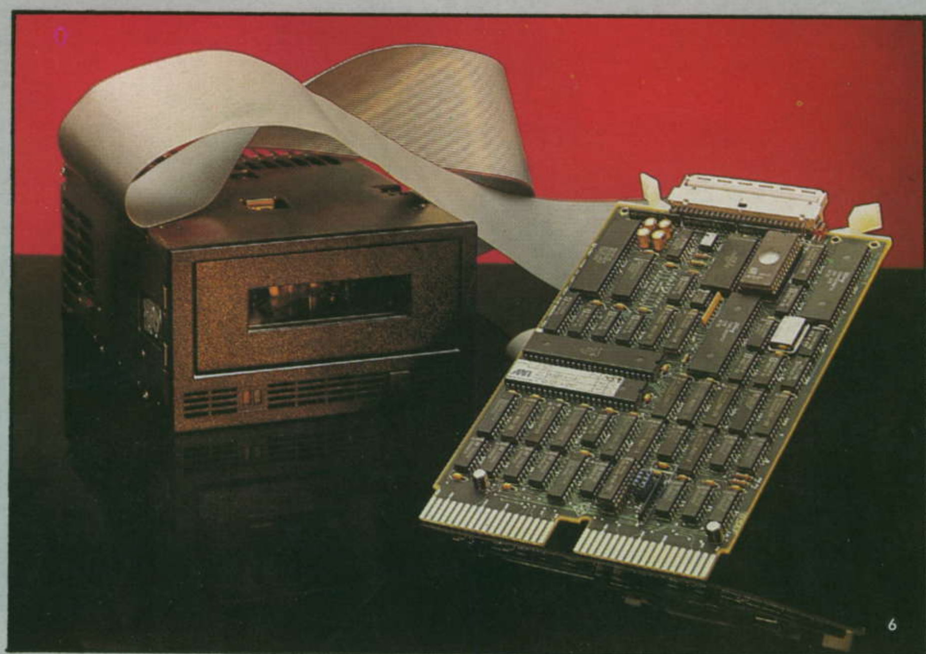
***Bild 8** Single Board Computer SBC J11-8 von ROI; J11-Mikroprozessor von DEC, 32-KByte-Cache, 8 serielle Schnittstellen V.24/IFSS, Q22-Bus, voll LSI-11-kompatibel*







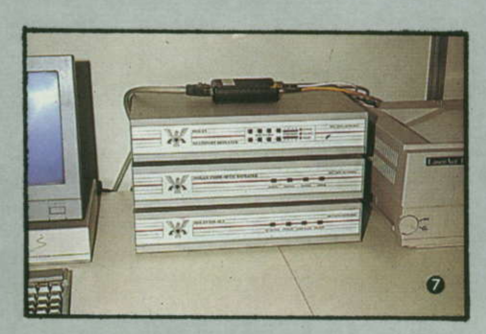


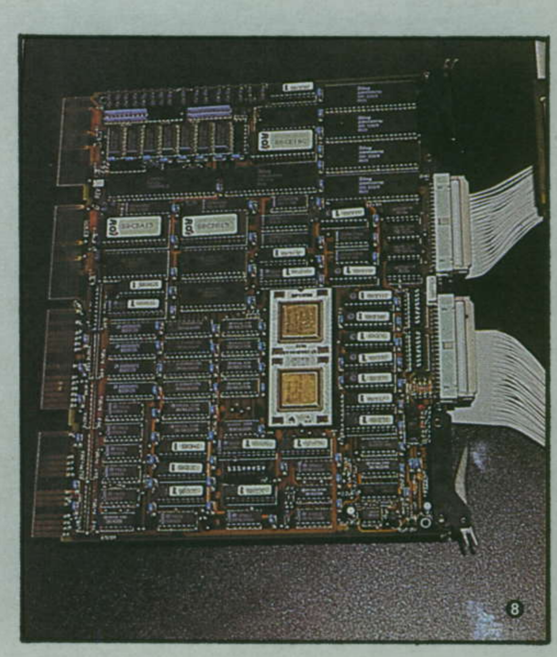






INTERDECK '89

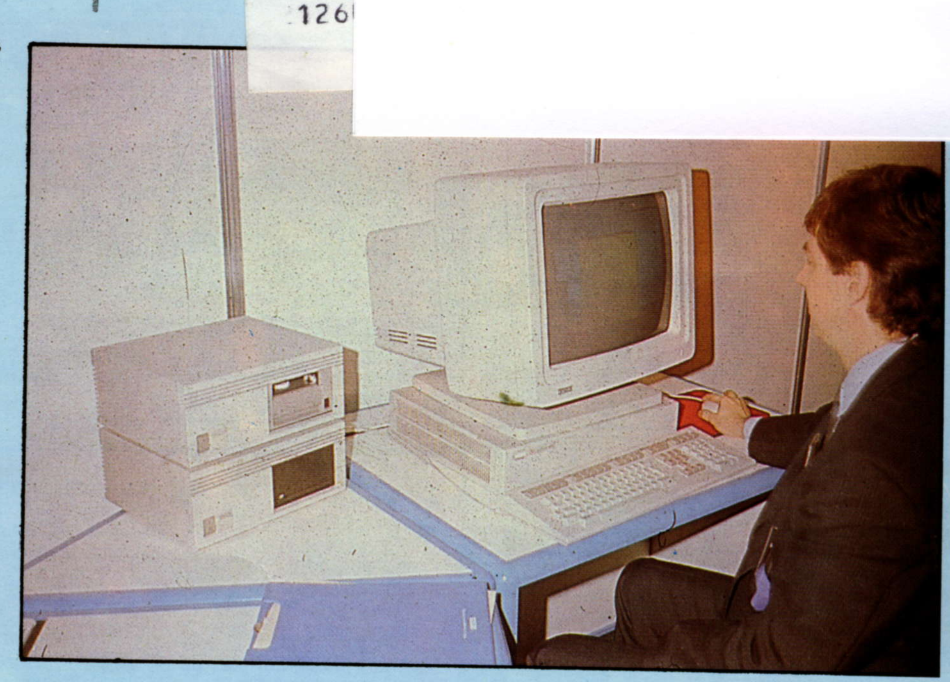




ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 9
S. 257–288

vorgestellt

RISC-Workstation jetzt auch von DEC



DECstation 3100 mit monochromatischem 19-Zoll-Bildschirm und Zusatzlaufwerk RZ55



DECstation 3100 mit zusätzlichem Kassettenmagnetbandlaufwerk TK50Z

Anfang des Jahres kündigte die Fa. Digital Equipment Corporation (DEC) ihre erste RISC-Workstation an, die die Palette der bewährten VAXstation-Familie (VAXstation 2000, VAXstation II/GPX, VAXstation 3200, VAXstation 35xx) um ein neues Modell an der oberen Leistungsgrenze ergänzt, wobei das Preis-/Leistungsverhältnis gegenüber den bisherigen Workstations um 50 bis 100 Prozent verbessert werden konnte.

Die neue **DECstation 3100** wurde auf der Basis des RISC-Prozessors der MIPS-Corporation R 2000, mit 14 MIPS (Millionen Befehle pro Sekunde) nach dem Dhrystone-Benchmark bei einem internen Takt von 16,67 MHz, realisiert. Durch die Gleitkommaeinheit R 2010 wird bei einfacher Genauigkeit eine Verarbeitungsleistung von 4 MFLOPS (Millionen Gleitkommaoperationen pro Sekunde) bzw. bei doppelter Genauigkeit 2,1 MFLOPS erreicht. Der Hauptspeicher kann von 8 bis 24 MByte ausgebaut werden. Zur Ausnutzung der vollen Leistungsfähigkeit des Prozessors sind zwei Cachespeicher zu je 64 KByte für die Daten und Befehle realisiert.

Erstmalig hat DEC auch den Industriestandard-Bus SCSI (Small Computer System Interconnect) mit einer Übertragungsleistung von 4 MByte implementiert. An SCSI können maximal zwei ausgebaute 3¹/₂-Zoll-Winchesterlaufwerke RZ23 (104 MByte, 23 ms mittlere Zugriffszeit) angeschlossen werden. In externen Erweiterungsboxen sind maximal vier 5¹/₄-Zoll-Winchesterlaufwerke RZ55 (332 MByte, 24 ms mittlere Zugriffszeit) und das Kassettenmagnetbandlaufwerk TK50Z (95 Byte) möglich. Für den LAN-Anschluß ist ein Ethernet-Anschluß (Thin-Wire) vorgesehen.

Für die interaktive grafische Arbeit stehen ein 15-Zoll- oder ein 19-Zoll-Bildschirm (monochromatisch oder color) mit einer Auflösung von 1024 × 864 Pixel zur Verfügung. Bei der Color-Variante sind 256 Farben aus einer Menge von 16,7 Mio Farbtönen möglich.

Auf Grund des Einsatzes eines RISC-Prozessors ist die DECstation 3100 zu den VAXstations inkompatibel. Deshalb steht das Standardbetriebssystem VAX/VMX nicht zur Verfügung, und es mußte auf ULTRIX, das UNIX-kompatible DEC-Betriebssystem, orientiert werden. Als Programmiersprachen werden vorläufig nur FORTRAN-77 und C geliefert.

Als ein ULTRIX-Server für LAN wird weiterhin die **DECstation 3100S** mit 1 GByte Plattenspeicherkapazität angeboten, während in Ergänzung dazu auf gleicher Basis das **DECsystem 3100** ein UNIX-Mehrnutzersystem mit 8 bis 64 Terminals darstellt.

Im Vergleich zu den VAX-Rechenanlagen kann die neue Workstation mit ca. 12facher Leistung einer VAX-11/780 bewertet werden. Da diese neuen Systeme vor allem in den UNIX-Markt vorstoßen, konnten in den ersten 6 Monaten bereits über 10 000 Workstations umgesetzt werden.

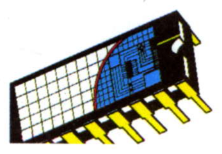
In der Zwischenzeit wurde von Sun Microsystems auch eine RISC-Workstation Sparcstation 1 vorgestellt, die je nach Anwendungsfall eine Leistung von 60 bis 90 Prozent der DECstation 3100 haben soll und damit deutlich unter der Leistung der DECstation 3100 liegt.

Von DEC wird beabsichtigt, noch in diesem Jahr eine leistungsgesteigerte Workstation mit dem wesentlich schnelleren RISC-Prozessor R 3000 der MIPS Corporation auf den Markt zu bringen.

Prof. Dr. Th. Horn



ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Grund
Heft 10 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
1-Megabit-Speicher
U 61000
CMOS-Mikroprozessorsystem
Hybridschaltkreise



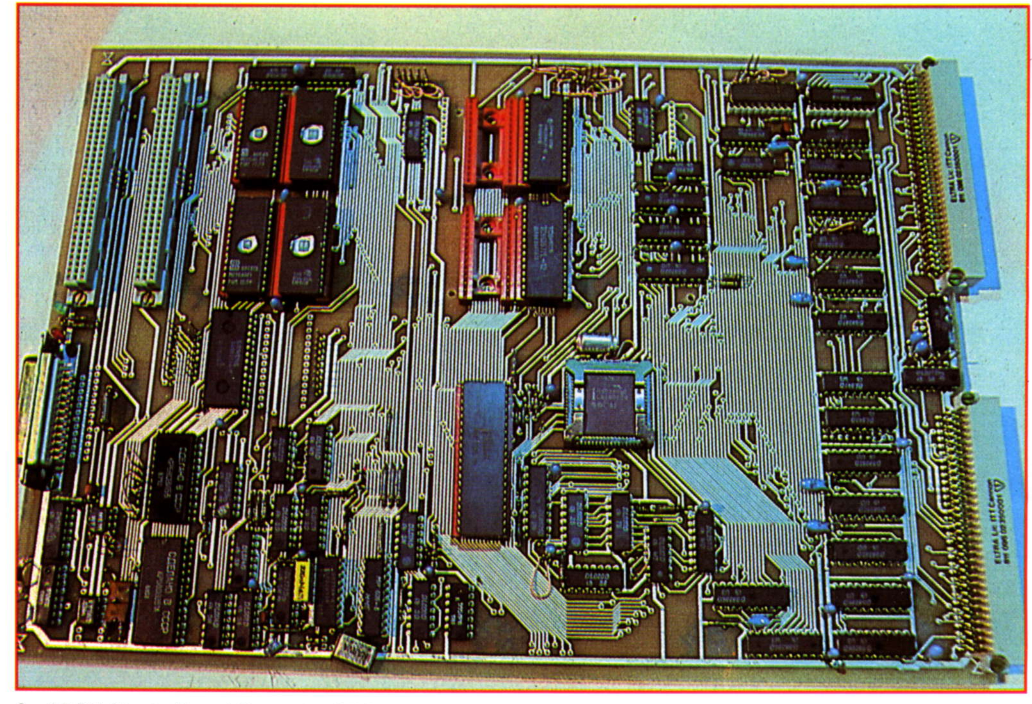
13. Mikroelektronik-Bauelementesymposium

Das dreizehnte Bauelementesymposium mit dem Schwerpunkt auf der Mikroelektronik wurde traditionsgemäß im Mai in Frankfurt(Oder) ausgerichtet. Dafür wurde wiederum das im Bild 1 gezeigte Sport- und Ausstellungszentrum am Westkreuz genutzt. Veranstalter waren das Kombinat Mikroelektronik und die Kammer der Technik Frankfurt(Oder). Das Symposium stand unter der Schirmherrschaft des Ministers für Elektrotechnik und Elektronik, Felix Meier. Den 2400 Teilnehmern wurde nach den Plenarvorträgen des Ministers für Elektrotechnik und Elektronik und der Kombinate Mikroelektronik, Carl Zeiss JENA, Robotron, Automatisierungsanlagenbau und Nachrichtenelektronik ein Fachvortragsprogramm mit 56 Vorträgen über neu entwickelte Bauelemente und deren Applikation geboten.

Auf dem Bauelementesymposium nimmt der Erfahrungsaustausch einen immer größeren Stellenwert ein. Die 1200 m² große Ausstellungsfläche mit 850 vorgestellten Bauelementen und 195 Exponaten der Anwenderindustrie sowie der Akademien und Hochschulen, die Podiumsgespräche und die Posterdiskussionen boten dazu ausreichende Möglichkeiten.



1



2 *16-Bit-Single-Board-Computer SBC-WPU-80601*

Von besonderem Interesse waren (die zur Leipziger Frühjahrsmesse erstmals gezeigten) Schaltkreise der Mikroprozessorsysteme U 80600 und U 84C00, der 1-MBit-DRAM U 61000 sowie die anwendungsspezifischen Schaltkreissysteme U 5300 und U 1600. Der 8-KByte-SRAM U 6264, der 8-KByte-EPROM U 2764 sowie die schnelle CMOS-Logikfamilie U 74HCT wurden vorgestellt und Erfahrungen beim Einsatz von oberflächenmontierbaren Bauelementen ermittelt.

Über das 16-Bit-Mikroprozessorsystem U 80600 informierten wir Sie bereits ausführlich in MP 5/1989, den 1-MBit-DRAM U 61000 und den 8-Bit-CMOS-Prozessor U 84C00 stellen wir in diesem Heft näher vor.

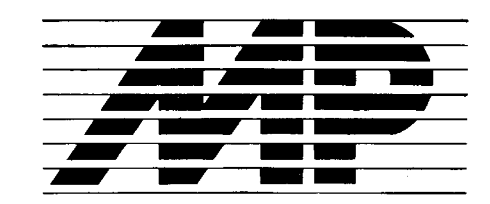
Für Nutzer des EC 1834 dürfte die Ankündigung einer Beschleunigerkarte auf der Basis des U 80601 von Interesse sein. Die **BK 600** – auch Turbo-Karte genannt – ermöglicht eine Beschleunigung der Rechengeschwindigkeit des EC 1834 auf das 3- bis 4fache. Bei Koprozessoranwendung ist sogar ein Faktor von 5 bis 8 möglich. Die Karte stellt ein vollständiges Subsystem dar, das im EC 1834 an die Stelle der RAM-Erweiterung gesteckt werden kann. Sie übernimmt die Abarbeitung aller Betriebssystem-, Treiber- und Anwenderprogramme, wobei das ursprüngliche Prozessorsystem als Slavesystem weiterverwendet wird. Damit wird der Zugriff auf alle Systemressourcen gewährleistet. Der RAM-Bereich des EC 1834 kann hierbei als Disk-Cache für Hard- und Floppydisks verwendet werden. Die Bauelementebasis für die Beschleunigerkarte, die eine Gemeinschaftsarbeit der Kombinate Mikroelektronik und Robotron darstellt, sind die Schaltkreise U 80601, U 80613, DS 82284, U 80606, U 41256-15, U 214 D, U 6514 D sowie die DL- und DS-Logikreihe.

Von den in der Ausstellung gezeigten Exponaten haben wir für Sie einige ausgewählt, die wir Ihnen nachfolgend kurz vorstellen wollen.

Die Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, zeigte den 16-Bit-Single-Board-Computer **SBC-WPU-80601** (Bild 2). Der SBC in DKL-Technik mit den Abmessungen 233 mm × 330 mm wurde mit folgenden Schaltkreisen vorgestellt: CPU 80286, Taktgenerator 82284, zwei Buscontroller 82288, Busarbiter 82289, Arithmetikprozessor 80287, zwei Interruptcontroller 8259A, V.24-/IFSS-Interface mit USART 8251A, Timer 8253, vier statische RAMs 6264 ... 62256 sowie vier EPROMs 2764 ... 27256. Der SBC-WPU-80601 bildet das Kernstück eines modularen Systems mit MMS-16-Global- und Lokalbus sowie SBX-Bus mit SCSI-Adapter. Weiterhin wurden systemeigene Ergänzungen vorgestellt: Grafikmodul mit 82720 und MMS-16-Bus, Winchestercontroller mit 82062 und SBX-Bus. In Funktion wurde die Konsolenkopplung des SBC mit SCSI-Interface vorgeführt. Die Software besteht aus dem 957B-Monitor mit Erweiterungen, SCP und DCP im Real Address Mode der CPU. In Beispielen wurden die Funktionsprinzipien des Protected Mode gezeigt. Der Grafikmodul erlaubt eine Darstellung von 800 × 800 Pixeln, ein Demo auf

Fortsetzung auf der 3. US

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 10 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR–1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

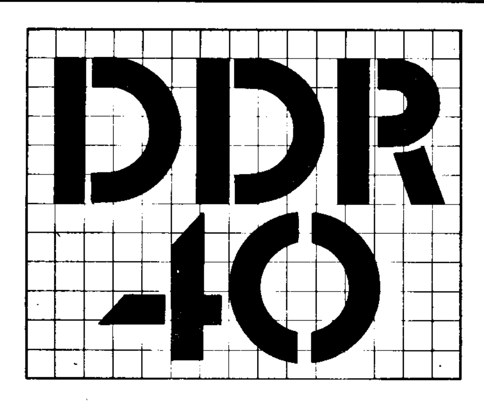
Redaktionsschluß 15. August 1989

AN (EDV) 49837

Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propagandit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR–7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR–7010 Leipzig



40 Jahre DDR – das bedeutet kontinuierliche Entwicklung unseres Staates und seit langem die Einheit von Wirtschafts- und Sozialpolitik. Folgerichtig war 1976 der Entschluß zur Entwicklung und breiten Anwendung der Mikroelektronik gefaßt worden, und die in unserer Zeitschrift bisher veröffentlichten Beiträge sind ein Ausdruck für die erfolgreiche Verwirklichung dieser Vorhaben. In unserem Mai-Heft konnten wir Sie beispielsweise umfassend über das neue, schnelle 16-Bit-Mikroprozessorsystem U 80600 aus dem VEB Kombinat Mikroelektronik Erfurt informieren, dessen Serienfertigung zu Ehren des 40. Jahrestages gegenwärtig beginnt.

In den beiden ersten Beiträgen dieses Heftes (**Seite 291** bzw. **Seite 292**) erläutern wir die Bedeutung der Megabitspeicher und stellen den 1-MBit-DRAM U 61000 ausführlich vor. Mit der Entwicklung dieses Speicherschaltkreises haben die Werktätigen des VEB Carl Zeiss JENA Leistungen vollbracht, zu denen gegenwärtig nur wenige führende Länder bzw. Hersteller in der Welt in der Lage sind. Bekanntlich waren bereits im vergangenen Jahr erste Muster präsentiert worden, und zum diesjährigen Geburtstag der Republik kann die Aufnahme der Pilotproduktion abgerechnet werden.

Neben dem 16-Bit-Mikroprozessorsystem U 80600 zeigte das Kombinat Mikroelektronik zur Leipziger Frühjahrsmesse als weitere Neuheit sein erstes 8-Bit-System in der stromsparenden CMOS-Technik. Wir stellen Ihnen das System U 84C00 in diesem Heft ab **Seite 296** näher vor.

Ebenfalls zur diesjährigen Frühjahrsmesse offerierte das Kombinat Keramische Werke Hermsdorf in Leipzig Muster von 4-Megabitspeichern auf Hybridschaltkreisträgern. Sie bestehen, je nach Variante, aus jeweils 16 Chips des 256-KBit-DRAMs U 61256 oder 4 Chips des U 61000 aus dem Kombinat Carl Zeiss JENA. In dem Beitrag auf **Seite 298** wird ein Überblick über den Stand der Fertigung von Hybridschaltkreisen im Kombinat Keramische Werke Hermsdorf gegeben, der diese Speicherschaltkreise einschließt.

Vorschau

Für Heft 11 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Von Turbo 3.0 zu Turbo 5.0
- Multiprozessorsysteme mit U 8000
- Druckerinitialisierung
- DOS-Gerätetreiber für SCOM-LAN

Inhalt

Gute Bilanz bei Schlüsseltechnologien im 40. Jahr der DDR

In Verwirklichung der ökonomischen Strategie der SED wurde das hohe Tempo der Einführung von Schlüsseltechnologien fortgesetzt. Das ergibt sich aus der Mitteilung der Staatlichen Zentralverwaltung für Statistik über die Durchführung des Volkswirtschaftsplanes 1989 im ersten Halbjahr. Mit der steigenden Produktion von höchstintegrierten Speicherschaltkreisen wurde die Bereitstellung elektronischer Bauelemente für die Rechen- und Automatisierungstechnik weiter verbessert. Ausschlaggebend für die breitenwirksame Durchsetzung der Schlüsseltechnologien in allen Bereichen der Volkswirtschaft sind die hohen Steigerungen der Produktion von solchen Erzeugnissen wie unipolaren monolithisch-integrierten Schaltkreisen um 48 Prozent, bipolaren monolithisch-integrierten Schaltkreisen um 22 Prozent, Druckern für Computer um 30 Prozent, Büro- und Personalcomputern um 15 Prozent und Industrierobotern um 10 Prozent.

- Die rechentechnische Basis der Volkswirtschaft ist mit der Produktion von 31 297 Büro- und Personalcomputern weiter gestärkt worden. Wesentlich gesteigert wurde die Fertigung von 16-Bit-Computern mit größerer Rechengeschwindigkeit und erhöhter Speicherkapazität.
- Im ersten Halbjahr 1989 sind 9100 CAD/CAM-Arbeitsstationen und -Systeme neu in Betrieb genommen worden. Sie wurden vorrangig in der Forschung und Entwicklung, der Konstruktion und der Projektierung, zur Softwareproduktion sowie für die Produktionsvorbereitung und -lenkung angewendet.

Gegenwärtig sind in der Volkswirtschaft rund 82500 CAD/CAM-Arbeitsstationen und -Systeme im Einsatz. Das Angebot an leistungsfähigen Ausrüstungen für die CAD/CAM-Technik ist durch die Steigerung der Produktion von Kleindatenverarbeitungsanlagen um 51 Prozent – insbesondere mit 32 Bit Verarbeitungsbreite – erweitert worden.

- Die Softwareherstellung nahm um 12 Prozent zu.
- Ende des ersten Halbjahres 1989 waren insgesamt über 96000 Industrieroboter in der Volkswirtschaft eingesetzt. Fortgesetzt wurde die Einführung flexibler automatisierter Fertigungssysteme.

Im Bereich Elektrotechnik und Elektronik wurde ein überdurchschnittlicher Leistungs- und Effektivitätszuwachs zur beschleunigten Entwicklung und Anwendung der Mikroelektronik in der Volkswirtschaft erreicht. Hervorragenden Anteil daran hatten mit hohen Steigerungsraten der Arbeitsproduktivität die Kombinate Robotron Dresden mit 23 Prozent, Mikroelektronik Erfurt mit 14 Prozent, Elektronische Bauelemente Teltow mit 12 Prozent und Carl Zeiss JENA mit 11 Prozent. Mit der wachsenden Produktion von höchstintegrierten Speicherschaltkreisen verbesserte sich die Bereitstellung elektronischer Bauelemente für die Rechen- und Automatisierungstechnik.

Mit dem Abschluß der Entwicklung des Bildungscomputers A 5105 im Kombinat Robotron wird das Erzeugnisspektrum der Rechentechnik durch einen leistungsfähigen und kostengünstigen Rechner ergänzt, der sowohl für die Ausbildung als auch für viele andere Aufgaben eingesetzt werden kann. *MP*

Hohe Aufgaben für Sömmerdaer Werktätige

Die Produktion von Seriendruckern soll in diesem Jahr im Büromaschinenwerk „Ernst Thälmann" Sömmerda um nahezu ein Drittel gegenüber 1988 steigen. Wichtiges Wettbewerbsvorhaben ist außerdem die Verdopplung der Stückzahl leistungsfähiger 16-Bit-Computer vom Typ EC 1834.

Vor den 13000 Werktätigen des Werkes steht die Aufgabe, mit dem Plan 1990 höhere Zuwachsraten anzugehen als in den Jahren zuvor. So soll die Nettoproduktion in dem größten Robotron-Kombinatsbetrieb um 28,6 Prozent steigen, die Arbeitsproduktivität in der gleichen Größenordnung. 32000 Personalcomputer, die meisten davon mit 16 Bit Verarbeitungsbreite, und 165000 Seriendrucker werden – so lautet das Ziel – im nächsten Jahr den Betrieb verlassen. *ADN/ND*

42. Tagung der Paritätischen Regierungskommission

Im Mai fand in Moskau die 42. Tagung der Paritätischen Regierungskommission für ökonomische und wissenschaftlich-technische Zusammenarbeit zwischen der DDR und der UdSSR statt.

Die Kommission behandelte unter anderem die von den Ministerien, Kombinaten, Vereinigungen und Betrieben in Auswertung der DDR-Ausstellung in Moskau abgestimmten Maßnahmen, die darauf gerichtet sind, die ökonomische und wissenschaftlich-technische Zusammenarbeit zwischen beiden Ländern weiter zu entwickeln und zu vertiefen. An erster Stelle steht dabei, Spitzenleistungen in Forschung und Produktion, insbesondere bei den Schlüsseltechnologien, zu erzielen sowie eine hohe Effektivität auf dem Weg der Intensivierung und durch Spezialisierung und Kooperation zu erreichen.

Auf dem Gebiet der Mikroelektronik als Hauptkomplex der Zusammenarbeit zwischen der DDR und der UdSSR wurden gemeinsame Vorhaben zur Sicherung eines hohen Technologieniveaus beraten. So stand auf einem Treffen des Ministers für Elektrotechnik und Elektronik der DDR, Felix Meier, mit dem Minister für elektronische Industrie der UdSSR, Wladislaw Kolesnikow, die weitere Zusammenarbeit in der Mikroelektronik für den Zeitraum des kommenden Fünfjahrplanes im Mittelpunkt. Die gemeinsame Arbeit richtet sich vor allem auf Technologien zur Herstellung von 1- und 4-Megabit-Schaltkreisen und zur Kooperation der Länder im Elektronikmaschinenbau. Vereinbart wurde die Vorbereitung neuer beziehungsweise die Verlängerung bestehender Regierungsabkommen auf den Gebieten der Mikroelektronik, passiven elektronischen Bauelemente, elektronischen Vermittlungstechnik und Lichtleiterübertragungstechnik.

Mit dem Vizepräsidenten der Akademie der Wissenschaften der UdSSR, Jewgeni Welichow, wurde ein Meinungsaustausch geführt über Möglichkeiten der Zusammenarbeit auf dem Gebiet der Mikroprozessorsysteme, die von großer Bedeutung für die Rechentechnik und für moderne Automatisierungslösungen im Maschinenbau sind. *ADN*

Erste Funktionsmuster übergeben

Der 32-Bit-Mikroprozessor für den „kleinen Würfel"

Nach knapp 3jähriger Entwicklungszeit wurden am 17. Juli dieses Jahres die ersten Muster des 32-Bit-Mikroprozessors U 80701 – des wichtigsten Schaltkreises eines neuen Mikroprozessorsystems – im Forschungszentrum des Kombinates Mikroelektronik Erfurt hergestellt.

Am 14. August übergab eine Abordnung der über 250 daran beteiligten Forscher und Entwickler aus Anlaß des 40. Jahrestages der DDR im Hause des Zentralkomitees der SED die ersten Schaltkreise dieses Mikroprozessors an den Generalsekretär des ZK der SED, Erich Honecker. Damit löste das Erfurter Forschungszentrum eine Verpflichtung ein, die im Wettbewerb zu Ehren des 40. Jahrestages – anläßlich des Besuches von Erich Honecker im VEB Mikroelektronik „Karl Marx" Erfurt am 22. Mai 1986 – übernommen wurde. Beteiligt an dieser Entwicklung waren auch Kollektive des Kombinates Robotron und der Akademie der Wissenschaften der DDR.

Während des Treffens nannte der Generaldirektor des Kombinates Mikroelektronik, Prof. Dr. Heinz Wedler, die Vorzüge dieses ersten 32-Bit-Mikroprozessorsystems der DDR. Gegenüber dem bisher vom Kombinat Robotron produzierten 32-Bit-Rechner K 1840 (siehe MP 10/1988, Seite 311), der etwa 10000 integrierte Schaltkreise besitzt, benötige der mit dem neuen Mikroprozessorsystem ausgestattete Rechner nur noch etwa 900 Schaltkreise. Sind für den K 1840 noch 15 Kilowatt Anschlußleistung erforderlich, so soll der nachfolgende Rechner nur noch 1,5 Kilowatt verbrauchen. Für die Erfüllung der Politbürobeschlüsse zur Mikroelektronik sei die materiell-technische Basis in Erfurt-Südost durch den Bau der nunmehr drei modernen Chipfabriken geschaffen worden. Da jedoch das Forschungszentrum erst Anfang der neunziger Jahre fertiggestellt sein wird, mußten die Entwicklungsarbeiten in den neugebauten Fabriken parallel zur täglichen Produktion organisiert werden. Die Mikroelektroniker haben sich nun vorgenommen, den Entwicklungsabschluß des 32-Bit-Mikroprozessorsystems, das insgesamt 12 Schaltkreistypen umfaßt, mit einem hohen Tempo anzugehen. Im Schaltkreisentwurf konzentrieren sie bereits jetzt die Kräfte auf das folgende Mikroprozessorsystem, das drei- bis fünfmal leistungsfähiger sein wird als das gerade vorliegende.

Die technischen Daten des U 80701 können sich sehen lassen. So sind die 130000 Transistoren auf einem Chip mit einer Fläche von 84 mm^2 in einem 68poligen Gehäuse untergebracht. Internationale Spitzenwerte werden mit der Taktfrequenz von maximal 40 MHz erreicht. Die physisch adressierbare Speichergröße ist 16 MByte, während virtuell 4 GByte adressierbar sind. Der U 80701 wird in dem neuen 32-Bit-Rechner K 1820 eine Verarbeitungsleistung von einer Million Operationen pro Sekunde ermöglichen.

Der 1. Stellvertreter des Generaldirektors des Kombinates Robotron, Dr. Dieter Walter, war als Projektleiter für den K 1820 ebenfalls bei der Übergabe anwesend. In seinen Ausführungen stellte er unter anderem fest: „Unseren großen K 1840, auf den wir vor zwei Jahren fürchterlich stolz waren, werden wir als kleinen Würfel an den Arbeitsplatz bringen und damit natürlich den Konstrukteuren und Entwerfern ganz andere Möglichkeiten erschließen."

Erich Honecker betonte, daß die DDR mit diesem Leistungsstand nachgewiesen habe, daß sie mit der seit dem XI. Parteitag geschaffenen Entwicklungs- und Produktionsbasis in der Lage sei, leistungsfähige 32-Bit-Mikroprozessorsysteme nach eigenen Entwurfsmethoden zu entwickeln und herzustellen. *MP*

Produktionsbeginn des U 61000

Über die Messeofferte sowie die Entwicklung des VEB Carl Zeiss JENA informierte der Stellvertreter des Generaldirektors, Wolfgang Nordwig, auf einer internationalen Pressekonferenz anläßlich der Leipziger Herbstmesse 1989. Unter anderem erläuterte er, daß die aus der Entwicklung der Mikroelektronik im Kombinat gewonnenen Erkenntnisse zunehmend den technischen Konsumgütern zugute kommen. Mit Interesse wurde die Bestätigung aufgenommen, daß in Dresden die Produktion des 1-MBit-Speicherschaltkreises begonnen habe. *MP*

Rank-Xerox-Ausstellung

Rank Xerox hat sich entschlossen, den nach eigenen Angaben meistverkauften Kopierer der Welt, den auch in der DDR verbreiteten Xerox 1025Z, aufgrund der Nachfrage bis 1991 weiterzuproduzieren. Gleichzeitig wurde mitgeteilt, daß zwar die Ersatzteilversorgung für ältere Modelle ausläuft, für die Modelle 7000 und 3107 jedoch Ersatzteile noch bis 1991, Toner, Entwickler und Trommeln noch bis 1992 zur Verfügung stehen werden.

Um potentiellen Kunden noch vor der Leipziger Frühjahrsmesse 1990 die Geräte der neuen 50er Serie vorstellen zu können, wird die Firma im November in Berlin eine Ausstellung veranstalten. Weitere Auskünfte erteilt: Rank Xerox Limited, Internationales Handelszentrum, Friedrichstraße, Berlin, 1086; Tel. 20 96 26 75. *MP*

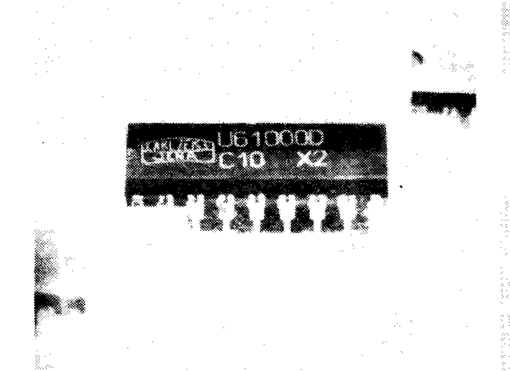
Der Megabitspeicher und die Mikroprozessortechnik

Prof. Dr. Bernd Junghans,
VEB Forschungszentrum Mikroelektronik Dresden
Betrieb des Kombinates VEB Carl Zeiss JENA

Weshalb richtet sich in so ungewöhnlichem Maße die Aufmerksamkeit der Öffentlichkeit auf die Entwicklung von Halbleiterspeichern? Sind sie doch nur einer von Hunderten Bausteinen, die die Hardwarebasis der modernen Rechentechnik ausmachen. Gewiß sind sie ökonomisch wichtig, da sie bereits über einen langen Zeitraum und auch in der absehbaren Zukunft wertmäßig etwa 20 Prozent der Weltproduktion an integrierten Schaltkreisen ausmachen. Den Anwender dürften aber mehr die Gebrauchswerte interessieren. Und tatsächlich ist ein Großteil des in den letzten Jahren erzielten atemberaubenden Fortschritts der Mikroprozessortechnik durch den Einsatz immer höher integrierter Speicherbausteine ermöglicht worden. Hatten noch vor rund 10 Jahren lediglich Großrechner eine Hauptspeicherkapazität von einigen Megabyte, so ist diese Größe heute bereits Standard bei Arbeitsplatzcomputern. Das war nur möglich durch die Steigerung des Integrationsgrades der Speicherschaltkreise in diesem Zeitraum um etwa drei Größenordnungen.

So wurde Anfang der 80er Jahre der Hauptspeicher für die EDVA EC 1055 mit einer Kapazität von 1 Megabyte durch 9000 Stück 1-KBit-DRAM U 253 realisiert, wozu ein separater Schrank mit den Abmessungen von rund 0,7 m × 1,2 m × 1,8 m benötigt wurde. Es wäre weder ökonomisch erschwinglich, noch technisch sinnvoll, einem Personalcomputer einen solchen Schrank beizustellen. Die gleiche Speicherkapazität kann heute mit 9 Stück DRAM-Schaltkreisen U 61000 auf einer Leiterplattenfläche von rund 10 cm^2 untergebracht werden! Den durch die gewachsene Speichergröße möglich gewordenen enormen Gewinn an Rechnerleistung und Komfort wissen die Leser der „Mikroprozessortechnik" selbst einzuschätzen.

Weniger bekannt dagegen ist häufig sogar unter Informatikern die fundamentale Bedeutung der Speicherschaltkreise als Schrittmacher für die Technologieentwicklung, die aber letztlich allein die Sonderstellung der Speicher unter allen mikroelektronischen Entwicklungen ausmachen. Seit dem Eintritt in das Zeitalter der Großintegration (LSI) Ende der 60er Jahre haben die Speicher mit einer Verdopplung des Integrationsgrades aller 1,5 Jahre das Tempo des technologischen Fortschritts in der Mikroelektronik bestimmt. Dabei erstreckt sich das für die Speicherherstellung erarbeitete umfangreiche know-how nicht nur auf die Halbleitertechnologie an sich, sondern reicht von den einzusetzenden äußerst vielfältigen Chemikalien über die hochgezüchteten technologischen und meßtechnischen Ausrüstungen bis zur extremen Anforderungen genügenden Reinstraumtechnik und rechnergestützten Produktionssteuerung.



Bernd Junghans studierte von 1962 bis 1968 Elektronik am Moskauer Energetischen Institut. Danach war er wissenschaftlicher Aspirant und Oberassistent an der Sektion Physik/Elektronische Bauelemente der damaligen Technischen Hochschule (heute Technische Universität) Karl-Marx-Stadt. Seit 1976 arbeitet Bernd Junghans in unterschiedlichen Funktionen im Forschungszentrum Mikroelektronik Dresden, wo er sich mit Technologie- und Schaltkreisentwicklungen beschäftigt, zuletzt als Projektleiter des Megabitspeichers. Bernd Junghans ist Honorarprofessor an der Sektion Informationstechnik der Technischen Universität Dresden. Für seine Arbeiten wurde er zweimal mit dem Nationalpreis ausgezeichnet.

So war zum Beispiel für die CMOS-Technologie des Megabitspeichers /1/ die Beherrschung minimaler Strukturmaße um 1 Mikrometer nicht nur eine technologische Herausforderung an die Verfahrensentwickler, sondern ebenso an die Ausrüstungsentwickler, die unter anderem mit den Elektronenstrahlanlagen ZBA 21 und den optischen Strukturübertragungsanlagen AÜR des Kombinates VEB Carl Zeiss JENA entscheidende Vorleistungen erbringen mußten. Gleiches gilt für viele andere Betriebe. Hierin liegt eine der Katalysatorwirkungen der Mikroelektronik: Sie erfordert modernste Hochtechnologien in nahezu allen Bereichen der Volkswirtschaft und liefert zugleich entscheidende Voraussetzungen zur Erzielung derartiger Spitzenleistungen. Speicherschaltkreise eignen sich nun aufgrund ihrer logisch einfachen, regulären Struktur und ihrer außerordentlichen Empfindlichkeit gegenüber technologischen Schwankungen hervorragend als Testobjekte für die Entwicklung neuer Technologien und deren materiell-technischer Basis. Werden Speicherschaltkreise eines neuen technologischen Niveaus, das sich in der Regel vom Vorgängerniveau durch einen vierfach höheren Integrationsgrad und damit einhergehende Steigerung der Signalverarbeitungsgeschwindigkeit im Schaltkreis unterscheidet, mit akzeptabler Ausbeute beherrscht, kann auf dem soliden Fundament einer solchen **Basistechnologie** die Entwicklung eines breiten Sortimentes von Mikroprozessoren, Logikschaltkreisen, ASICs usw. erfolgen. In diesem Sinne war die Technologie des 256-KBit-DRAMs Basis für alle heute gebräuchlichen 32-Bit-Mikroprozessoren, während die Technologie des Megabitspeichers zu einer neuen Klasse von Mikroprozessoren führen wird, die die Leistungsfähigkeit heutiger Superrechner erreichen, wie das Beispiel des Prozessors 80860 der Firma Intel zeigt. Mit der weiteren Integrationsgradsteigerung, die mit den Etappen 4-MBit-DRAM, 16-MBit-DRAM und 64-MBit-DRAM (130 Mio Transistoren pro Chip!) bereits konzipiert ist, wird der Weg frei zur Integration kompletter elektronischer Systeme auf einem Chip. Künftig wird es für die Konkurrenzfähigkeit ganzer Industriezweige und Volkswirtschaften entscheidend sein, ob diese Technologie der Systemintegration beherrscht wird und verfügbar ist. Deshalb hat unsere Republik in den letzten Jahren große Anstrengungen unternommen, die Mikroelektroniktechnologie auf internationalem Spitzenniveau zu entwickeln. Beredter Ausdruck für die dabei erzielten Erfolge ist die Herstellung erster 1-Megabitspeicher zum 39. Jahrestag der DDR im Kombinat VEB Carl Zeiss JENA auf der Grundlage eigener Entwicklungsleistungen. Zum 40. Jahrestag kann bereits eine den wichtigsten Bedarf deckende Pilotproduktion dieses Schaltkreises abgerechnet werden, die eine gute Grundlage für weitere Schaltkreisentwicklungen in diesem technologischen Spitzenniveau bildet. Für die nächste Etappe, die mit dem Prototyp des 4-MBit-DRAMs ein Integrationsniveau von fast 10 Mio Transistoren pro Chip ermöglichen wird, laufen konzentrierte Arbeiten, um zum 41. Jahrestag unserer Republik die ersten funktionsfähigen Schaltkreise herstellen zu können.

Diese zielstrebige Entwicklung der Technologie der Höchstintegration ist ein entscheidender Faktor für die Zukunftssicherung unserer hochentwickelten Elektronikindustrie und darunter auch der Mikroprozessortechnik. Sie hat darüber hinaus Bedeutung für unsere gesamte Volkswirtschaft, wie E. Honecker feststellte: „Die weitere Entwicklung der DDR als modernes, leistungsfähiges Industrieland ist ohne die Mikroelektronik nicht vorstellbar." /2/

Literatur

/1/ Junghans, B.; Raab, M.: CSGT5 – eine moderne Basistechnologie für die Höchstintegration. Jenaer Rundschau 34 (1989) 1, S. 15
/2/ Honecker, E.: Aus dem Bericht des Politbüros an die 7. Tagung des ZK der SED. Dietz Verlag, Berlin 1988, S. 30

Der Megabitspeicher U 61000

Dr. Jens Knobloch, Andreas Scade
VEB Forschungszentrum Mikroelektronik Dresden

Überblick

Die größten zur Zeit produzierten Halbleiterspeicher mit wahlfreiem Zugriff (RAM = random access memory) sind dynamische Speicher. Diese RAMs werden mit der dynamischen Ein-Transistorzelle realisiert, weil alle anderen Zellkonzepte, beispielsweise statische Speicherzellen mit 4 oder 6 Transistoren, eine deutlich größere Fläche beanspruchen, was die Erzeugung gleichgroßer statischer Speicher im gleichen Technologieniveau verhindert.
Der dynamische Megabitspeicher (1-MBit-DRAM) U 61000 ist der erste DDR-Schaltkreis – und damit Technologietreiber – auf der Basis einer n-Wannen-CMOS-Technologie im Strukturniveau bis 1 µm (CSGT5d). Dieses Strukturniveau und die Anwendung von vier Leitbahnebenen (2 × Polysilizium und je 1 × Molybdänsilicid und Aluminium) erlauben eine sehr hohe Packungsdichte, so daß der Chip in kleine Standardgehäuse montiert werden kann. Die Herstellungstechnologie des Megabitspeichers ist in Tafel 1 charakterisiert. Auf der 4. Umschlagseite (oben) ist die Herstellung des Speichers in drei Bearbeitungsetappen dargestellt.

Tafel 1 Herstellungstechnologie CSGT5d

- 430 Teilschritte
- 18 fotolithografische Strukturierungen
- minimales Strukturraster: 2,5 µm
- Oxiddicke der Speicherkapazität: 12 nm
- Zellengröße: 3,8 × 9,0 = 34,2 μm^2
- Chipfläche: 5,1 × 12,85 = 65,5 mm^2
- Integrationsgrad: 2,3 Mio Bauelemente
- Eine Skalierung des Chips ist in Vorbereitung
- effektive minimale Kanallängen: 1 µm

Der U 61000 besitzt die Organisation 1048576 × 1 Bit und arbeitet im *Fast Page Mode* (FPM). Er ist vorwiegend für den Einsatz in der Rechen- und Nachrichtentechnik und in der Industrieelektronik vorgesehen und wird durch folgende Haupteigenschaften charakterisiert:

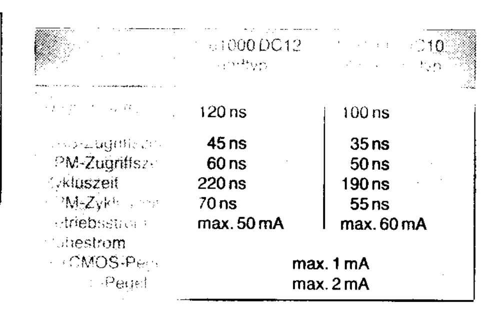
- dynamischer Schreib-/Lesespeicher mit wahlfreiem Zugriff
- hohe Arbeitsgeschwindigkeit und geringe Verlustleistung entsprechend Tafel 2
- TTL- und CMOS-Kompatibilität der Ein- und Ausgänge
- Tristate-Ausgangsstufen
- Betriebsspannung: 5 V ± 10 %
- 512 Refreshzyklen; Refreshzeit: 8 ms
- Betriebsarten im Normalbetrieb und im Fast Page Mode:
  READ CYCLE
  EARLY WRITE CYCLE
  READ MODIFY WRITE CYCLE
- Refreshmodi:
  $\overline{\text{RAS}}$ ONLY REFRESH
  $\overline{\text{CAS}}$ BEFORE $\overline{\text{RAS}}$ REFRESH
  HIDDEN REFRESH (READ und WRITE)
- Betriebstemperatur: 0 °C ... 70 °C.

Der Schaltkreis wird in einem 18poligen DIL-Plastgehäuse mit 7,62 mm Reihenabstand verkappt. Im Entwicklungszeitraum kann auch ein 18poliges Keramik-DIP-Gehäuse zum Einsatz kommen.
In diesem Fall lauten die Typenbezeichnungen
U 61000 CC12 und
U 61000 CC10.
Ein Gehäuse in Aufsetztechnik (SOJ 26/20) mit verringertem Platzbedarf ist in Vorbereitung.
Eine vollständige Übersicht über die Eigenschaften des Megabitspeichers U 61000 ist in der TGL 45536 (in Vorbereitung) enthalten.

Aufbau

Das speichernde Element der dynamischen Zelle ist ein Kondensator. Die dort gespeicherte Ladung repräsentiert die Information. Außer dem Kondensator enthält die Speicherzelle noch einen Auswahltransistor, über den die Information vom Kondensator auf die Bitleitung (Lesen) oder umgekehrt von der Bitleitung in den Kondensator (Schreiben) gelangen kann. Die Auswahl der Speicherzellen erfolgt durch ein überhöhtes Potential auf der Zeilen- oder Wortleitung.
Der U 61000 arbeitet mit einer planaren Ein-Transistorzelle, deren Schaltbild, Layout und Längsschnitt in Bild 1 dargestellt sind. Für höherintegrierte Speicher (z. B. 4-MBit-DRAMs) zeichnen sich neue Lösungen der Kondensatorherstellung ab, um bei weiterer Zellverkleinerung die Speicherkapazität in der notwendigen Größenordnung von zirka 40 fF zu erhalten. Da die Information als Ladung in einem Kondensator gespeichert wird, geht sie durch Leckströme mit der Zeit verloren. Man muß daher in sogenannten Refresh-Zyklen die Ladung regenerieren.

Tafel 2 Geschwindigkeits- und Leistungsparameter des U 61000



	[illegible]1000 DC12	[illegible]C10
[illegible]	120 ns	100 ns
[illegible]-Zugriffszeit	45 ns	35 ns
[illegible]PM-Zugriffsz[illegible]	60 ns	50 ns
[illegible]ykluszeit	220 ns	190 ns
[illegible]M-Zyk[illegible]	70 ns	55 ns
[illegible]triebsstr[illegible]	max. 50 mA	max. 60 mA
[illegible]uhestrom		
[illegible]CMOS-Pe[illegible]	max. 1 mA	
[illegible]-Pegel	max. 2 mA	

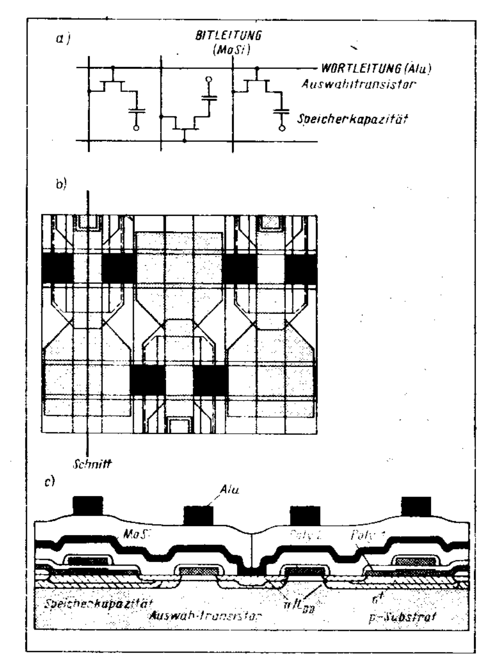


Bild 1 Speicherzelle U 61000
a) Schaltung b) Layout c) Längsschnitt

Jens Knobloch (45) studierte Regelungstechnik und war anschließend als wissenschaftlicher Assistent an der Technischen Universität Dresden tätig. 1974 promovierte er zum Thema „Entwurf bipolarer Assoziativspeicher".
Seit 1973 arbeitet er auf dem Gebiet des Entwurfs integrierter Schaltkreise im heutigen Forschungszentrum Mikroelektronik Dresden. Hier war er maßgeblich an der Entwicklung integrierter MOS-Speicher und deren Entwurfsgrundlagen beteiligt.
1988 verteidigte er die Promotion B „Entwurf und Entwurfstechnologie integrierter Schaltkreise". Jens Knobloch ist Themenleiter für die Entwicklung des 1-MBit-DRAMs. Für seine Arbeiten wurde er zweimal mit dem Nationalpreis ausgezeichnet.

Andreas Sc[illegible] (31) studierte v[illegible] bis 1983 an de[illegible] Technischen Un[illegible] Karl-Marx-Stadt [illegible] [illegible]mierte übe[illegible] Halbleiter.
Seit 1983 arbeitet er im Forschungszentrum Mikroelektronik Dresden als Spezialist für dynamische Speicher. Er war [illegible]nenleiter Schaltungsoptimierung für die Ausbeuteentwicklung des 64-KBit-DRAMs.
Bei der Entwicklung des 1-MBit-DRAMs ist er verantwortlicher Entw[illegible]ler der Gesamtschaltungskonzep[illegible] [illegible] Thе[illegible]ter für die Entwicklung des [illegible]eiterten Typspektrums des Megabitspeichers.

Die Speicherzellen sind auf dem Chip in Form einer Matrix angeordnet. Diese Speichermatrix wird in der Regel in Teilmatrizen zerlegt, deren Größe und Anordnung eine Optimierung der technologischen Größen Widerstand und Kapazität der Leitungen und der elektrischen Parameter Verzögerungszeit und Signalerkennung darstellt. Durch den Einsatz des Aluminiums kann die Wortleitung des 1-MBit-DRAMs zugunsten einer kurzen Bitleitung mit kleiner Kapazität und damit einer besseren Signalerkennung relativ lang ausgeführt werden. Die Matrixstruktur wird außerdem durch die Zielstellung einer internen 4-Bit-Organisation bestimmt, die aus Gründen der Leistungseinsparung, der Anwendung eines 4fach-Testmodus und des Einsatzes des gleichen Chips für einen Speicher mit der Organisation 256 K × 4 gewählt wurde. Das Foto auf der 4. Umschlagseite links unten zeigt die resultierende Matrixstruktur beim U 61000.
Die Ansteuerung der Matrix, die Taktabläufe und die Datenein- und Datenausgaben werden von *den* peripheren Schaltungen realisiert, die die Matrix umgeben. Das entsprechende Blockschaltbild ist in Bild 2 dargestellt. Über Bondinseln (siehe 4. Umschlagseite oben) und Schaltkreispins wird die Verbindung des Chips zur Leiterplatte hergestellt. Bild 3 zeigt die Anschlußbelegung des U 61000. Bedingt durch die Hinzunahme eines weiteren Adreßanschlusses A9 konnte das bis zum 256-KBit-DRAM übliche 16polige Gehäuse nicht beibehalten werden. Durch das 18polige Gehäuse bleibt ein freies Pin 4 übrig, welches nicht angeschlossen ist oder für einen besonderen Testmodus genutzt werden kann.

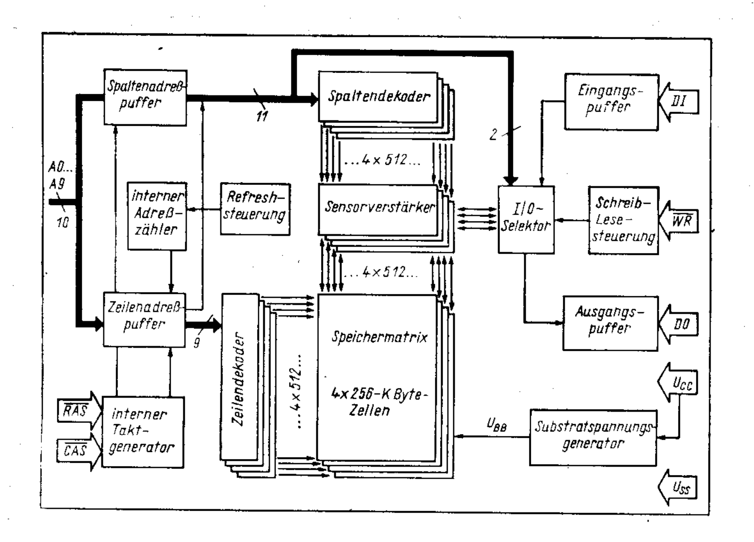
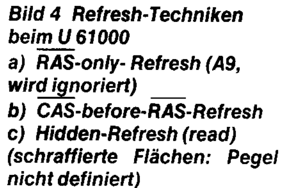


Bild 2 Blockschaltbild 1-MBit-DRAM U 61000



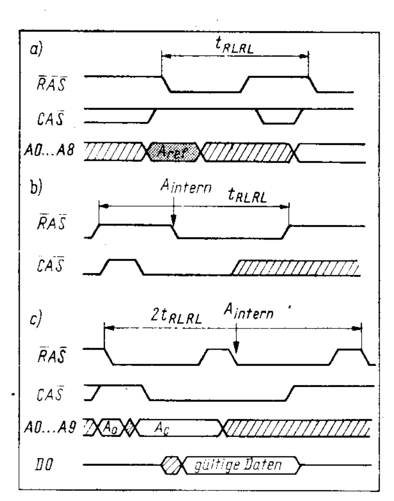


Bild 4 Refresh-Techniken beim U 61000
a) RAS-only-Refresh (A9, wird ignoriert)
b) CAS-before-RAS-Refresh
c) Hidden-Refresh (read) (schraffierte Flächen: Pegel nicht definiert)

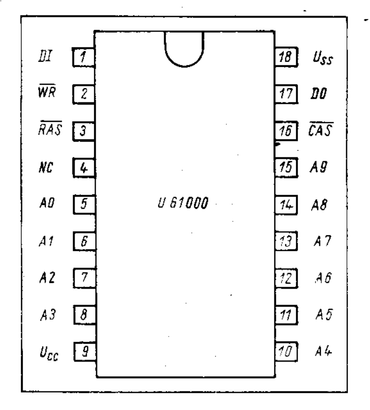


Bild 3 Anschlußbelegung des U 61000

A0...A9	**Adreßeingänge**	**RAS**	**Zeilenadreßstrobe**
WR	**Schreib-/ Lesesteuerung**	**CAS**	**Spaltenadreßstrobe**
DI	**Dateneingang**	**U_{cc}**	**Betriebsspannung**
DO	**Datenausgang**	**U_{ss}**	**Masse**
		NC	**nicht angeschlossen**

Arbeitsweise

Ein Megabitspeicher benötigt für die Auswahl der Speicherzelle eine 20-Bit-Adresse (2^{20} = 1 048 576). Um eine hohe Integrationsdichte solcher Speicher zu erhalten, werden kleine Gehäuse genutzt, die es verbieten, 20 Anschlüsse für die Adresse vorzusehen. Daher erfolgt die Übergabe der Adresse an den Speicherschaltkreis zeitmultiplex in zwei Hälften, der höherwertigen Zeilenadresse (row-address) und der niederwertigen Spaltenadresse (column-address) mit jeweils 10 Adressenbits. Die Übergabe dieser Teiladressen wird mit der High-Low-Flanke der entsprechenden Taktsignale RAS (row-address-strobe) und CAS (column-address-strobe) synchronisiert. Bei allen Speicheroperationen wird jeweils eine durch die Zeilenadresse bestimmte Wortleitung aktiviert, wodurch die angeschlossenen Speicherzellen ihre Ladungen an die jeweilige Bitleitung abgeben. Damit ist die Information in der Speicherzelle nicht mehr vorhanden, man spricht vom zerstörenden Lesen. Im Sensorverstärker wird am Bitleitungshub erkannt, ob eine Null oder eine Eins gespeichert war. Diese Information wird im Sensorregister statisch zwischengespeichert. Nach Abschluß dieser ersten Phase, die durch von RAS abgeleiteten Takten gesteuert wird, stehen beim Megabitspeicher 2048 Bit im Sensorregister zur Verfügung, die entsprechenden Zeilen in der Matrix bleiben für das spätere Rückschreiben aktiviert.

In der zweiten Phase wird dann durch die Dekodierung der Spaltenadresse ein Bit ausgewählt und ausgegeben (Read) oder verändert (Write) oder erst gelesen und danach neu geschrieben (Read Modify Write). Anschließend wird der Inhalt des Sensorregisters in die aktivierte Zeile zurückgeschrieben.

Mit der durch das Zeilenadreßbit A9 festgelegten Hälfte (1024 Bit) des Sensorregisters kann praktisch wie mit einem statischen Speicher gearbeitet werden, was im Fast Page Mode ausgenutzt wird.

Im Refreshzyklus (Auffrischungszyklus) entfällt die Manipulation an den einzelnen Bits, da hier durch internes Lesen, Verstärken und Rückschreiben jeweils zweier Zeilen nur die durch Leckströme veränderte Ladung der Speicherkondensatoren wiederhergestellt wird. Darum muß in diesem Fall keine Spaltenadresse bereitgestellt werden.

Auffrischen

Alle Zeilen des Megabitspeichers müssen innerhalb von 8 ms einmal aktiviert oder aufgefrischt werden, damit die Information in den Speicherkondensatoren auch unter ungünstigen Bedingungen (Grenzen von Betriebsspannung und Temperatur, fertigungsbedingte Toleranzen der Leckströme einzelner Zellen usw.) nicht verlorengeht. Dazu stehen drei Techniken zur Verfügung (Bild 4):

- RAS only refresh
- Hidden refresh
- CAS before RAS refresh.

Der **RAS-only-Refresh** ist die Standardform der Refreshsteuerung. Der Zyklus beginnt (wie jeder andere Speicherzyklus auch) mit einer aktiven RAS-Flanke (High-Low), zu der die passende Zeilenadresse an den Adreßeingängen bereitstehen muß. Der CAS-Eingang sollte inaktiv (High) bleiben, wodurch der Ausgang seinen hochohmigen Zustand behält und eine geringere Leistung verbraucht wird. Da nur 512 Refreshadressen erforderlich sind, wird das Adreßsignal A9 ignoriert.

Im Inneren vollzieht sich der schon beschriebene Ablauf des Auslesens, Verstärkens und Rückschreibens in der Zeile mit der angelegten Adresse. Dieser Vorgang muß innerhalb der Refreshzeit von 8 ms in jeder der 512 verschiedenen Refreshadressen mindestens einmal erfolgen, damit die Information nicht verlorengeht. Dies kann sowohl in gleichmäßigen Zeitabständen erfolgen, zum Beispiel durch Auffrischen jeweils einer Zeile nach zirka 15 μs, als auch im Burst-Betrieb, das heißt Auffrischen aller Zeilen unmittelbar hintereinander mit einer anschließenden längeren Phase ohne Refresh. Wenn sichergestellt ist, daß innerhalb der genannten Zeit in jeder Zeile mindestens ein normaler Speicherzugriff stattfindet, kann man auf einen zusätzlichen Refreshvorgang ganz verzichten. Der Zeitbedarf für das Auffrischen beträgt weniger als 1,5 % des normalen Speicherbetriebes.

Beim **Hidden-Refresh** und beim **CAS-before-RAS-Refresh** wird die benötigte Refreshadresse von einem chipinternen Adreßzähler bereitgestellt. Damit entfällt der Aufwand für deren externe Bereitstellung, und die Adreßeingänge können während dieser Refresharten beliebige Werte annehmen. Das Inkrementieren des internen Adreßzählers erfolgt nach jedem Zugriff.

Beim Hidden-Refresh kann man einen Refresh-Zyklus nach einem vorhergehenden Lesezyklus (oder Schreibzyklus) durchführen. Dazu bleibt das CAS-Signal am Speicher ununterbrochen aktiviert und damit die gelesene Information am Ausgang weiterhin gültig, während im Schaltkreis der Refresh abläuft. Das RAS-Signal muß dabei nach einem ordnungsgemäßen Lesezugriff zunächst inaktiviert und anschließend erneut aktiviert werden. Der Hidden-Refresh läuft also im Hintergrund in einem zusätzlichen Speicherzyklus ab.

Beim CAS-before-RAS-Refresh wird der interne Refreshvorgang durch eine geeignete Kombination der Signale RAS und CAS ausgelöst. Ist das CAS-Signal bei der RAS-Vorderflanke aktiv (Low), wird die im internen Adreßzähler vorgegebene Zeile aufgefrischt.

Zum Testen der ordnungsgemäßen Funktion des Adreßzählers kann im *Counter test cycle* mit der internen Refreshadresse – von außen ergänzt um die fehlenden Zeilenadresse – auf die Daten zugegriffen werden. Dieser Test muß mit acht CAS-befor-RAS-Zyklen zur Initialisierung des Adreßzählers eingeleitet werden. Anschließend werden zuerst bei festgehaltener Spaltenadresse in 512 Zyklen Daten in jede Zeile geschrieben. Diese Daten werden anschließend bei gleicher Spaltenadresse in 512 Zyklen wieder ausgelesen und getestet. Durch Wiederholen dieses Tests mit komplementären Daten wird die richtige Funktion des Adreßzählers gesichert.

Speicherzugriffe

Der Zugriff auf eine bestimmte Adresse erfor-

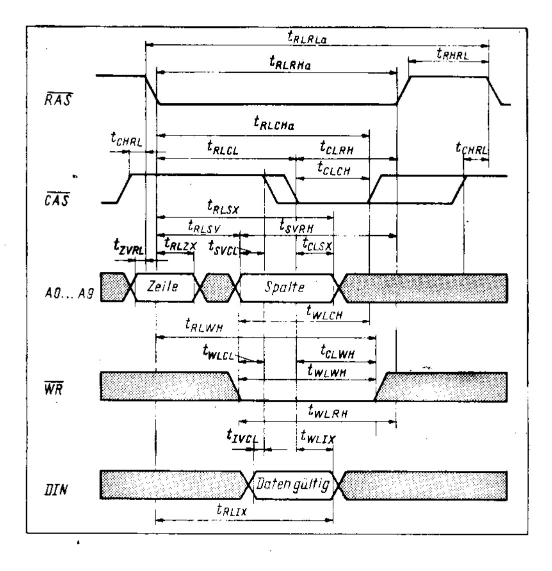


Bild 5 EARLY WRITE CYCLE (graue Flächen: Pegel nicht definiert)

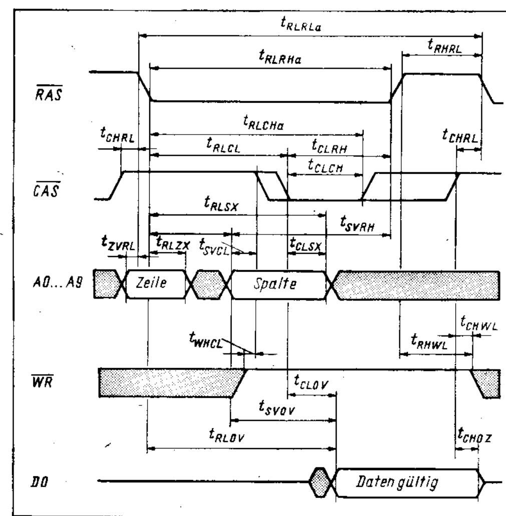


Bild 6 READ CYCLE (graue Flächen: Pegel nicht definiert)

dert im Gegensatz zum Refresh die Übertragung von beiden Adreßhälften, also Zeilen- und Spaltenadresse, die bei der aktiven Flanke (High-Low) des jeweiligen Strobe-Signals $\overline{RAS}$ oder $\overline{CAS}$ gültig sein müssen. Dabei entscheidet der Pegel des Lese-Schreibsteuerungssignals $\overline{WR}$ während der $\overline{CAS}$-Low-Flanke, ob es sich um einen Lese- oder einen Schreibzyklus handeln soll. Neben diesen beiden wichtigsten Betriebsarten gibt es bei den verschiedenen Typen der 1-MBit-DRAMs eine Reihe von typspezfischen Varianten, die eine andere Zugriffsorganisation mit höherer Arbeitsgeschwindigkeit beinhalten. Neben dem im U 61000 gewählten *Fast Page Mode* können das der *Static Column Mode* oder der *Nibble Mode* sein.
Alle im folgenden Abschnitt genannten Zeiten beziehen sich auf den U 61000 DC12.

Schreibzyklus

Bild 5 zeigt das Zeitdiagramm eines Schreibzugriffs. Das Schreibsignal $\overline{WR}$ ist während der aktiven $\overline{CAS}$-Flanke auf Low-Potential, und etwa zur gleichen Zeit muß die Information am Dateneingang DI gültig sein. Nach Ablauf der aktiven Low-Phase von $\overline{RAS}$ und $\overline{CAS}$ können diese Signale wieder inaktiv werden. Bevor der nächste Zyklus gestartet werden kann, braucht der Speicher etwas Zeit für interne Umladevorgänge (precharge time), weswegen die Zykluszeit bei dynamischen Speichern größer ist als die Zugriffszeit. Da bei diesem sogenannten Early-Write-Zyklus der Datenausgang DO hochohmig bleibt und im nachfolgend beschriebenen Lesezyklus die Information am Dateneingang DI beliebig sein kann, lassen sich beide Anschlüsse miteinander verbinden (Common I/O). Dies kann beim Anschluß des Speichers an den Datenbus eines Mikroprozessorsystems genutzt werden. Allerdings sind dann nicht mehr alle Sonderbetriebsarten des Speicherschaltkreises möglich.

Lesezyklus

Im Gegensatz zum Schreibzyklus muß beim Lesezyklus (Bild 6) das Schreibsignal während der gesamten $\overline{CAS}$-Low-Phase inaktiv bleiben. Von besonderem Interesse beim Lesezyklus sind die Zugriffzeiten. Wegen der zwei Strobe-Signale muß man zwischen den Zugriffszeiten von $\overline{RAS}$ und $\overline{CAS}$ unterscheiden. Außerdem wirkt unter bestimmten Bedingungen auch die Adreßzugriffszeit. Diese Bedingungen werden durch den Abstand der aktiven Flanken von $\overline{RAS}$ und $\overline{CAS}$ und durch den Vorhalt der Spaltenadresse vor der aktiven $\overline{CAS}$-Flanke bestimmt. Je nach Wahl dieser Zeiten wird die Zugriffszeit von $\overline{RAS}$, von $\overline{CAS}$ oder vom Adreßwechsel bestimmt.
Mit den spezifischen Werten des U 61000 DC12

$t_{RLOVmax}$ = 120 ns $t_{RLCLmin}$ = 25 ns
$t_{CLOVmax}$ = 45 ns $t_{SVCLmin}$ = 0 ns
$t_{SVOVmax}$ = 60 ns $t_{RLSVmin}$ = 20 ns

ergeben sich die in Bild 7 dargestellten Zugriffszeiten.
Mit dem Übergang von $\overline{CAS}$ in den inaktiven Zustand (High) gehen die Ausgänge wieder in den hochohmigen Zustand zurück, während $\overline{RAS}$ inaktiv werden kann, ohne daß die Ausgänge des Speichers ungültig werden (Hidden refresh). Die Zykluszeit im Schreib- und im Lesezyklus beträgt 220 ns. Geringere Zyklus- und Zugriffszeiten werden mit dem Fast Page Mode erreicht.

Read-Modify-Write-Zyklus

Die Grundidee eines Read-Modify-Write-Zyklus ist: In *einem* Zugriff auf eine Speicherzelle diese erst lesen und dann beschreiben. Wenn dabei der neue Speicherinhalt vom alten Inhalt abhängt, daß heißt, wenn die Schreibinformation aus den gelesenen Daten erzeugt wird, liegt ein echter Read-Modify-Write-Zyklus vor. Wie die Schreibdaten erzeugt werden, ist vom Speicher völlig unabhängig. Wird $\overline{WR}$ sofort nach Ablauf der Zugriffszeit aktiv, so wird eine Verarbeitung der gelesenen Daten nicht abgewartet, und

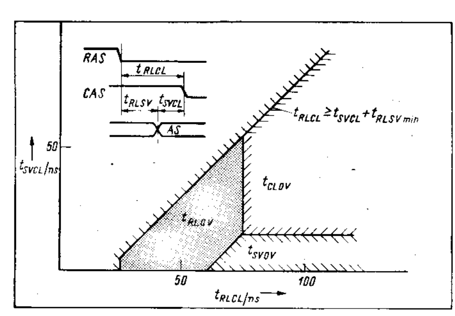


Bild 7 Wirksame Zugriffszeiten des U 61000 DC12 in Abhängigkeit von der $\overline{RAS}$-$\overline{CAS}$-Verzögerungszeit t_{RLCL} und der Spaltenadreßvorhaltzeit t_{SVCL}

die bereits an DI anliegenden Daten werden geschrieben. In diesem Fall spricht man von einem Read-Write-Zyklus, der sich im grundsätzlichen Ablauf vom Read-Modify-Write-Zyklus nicht unterscheidet.
Der zeitliche Ablauf eines Read-Modify-Write-Zyklus beginnt als normaler Lesezugriff, jedoch wird nach Ablauf der Zugriffszeiten das Schreibsignal $\overline{WR}$ aktiviert. Die dabei am Speichereingang DI liegende Information gelangt dadurch in die adressierte Zelle. Es ist klar, daß in einem echten Read-Modify-Write-Zyklus erst nach dem Lesen und einer Modifizierung der gelesenen Daten geschrieben werden kann.
Die Zykluszeit eines Read-Write-Zyklus beträgt 255 ns. Beim schon beschriebenen Early-Write-Zyklus bleibt der Speicheraus-

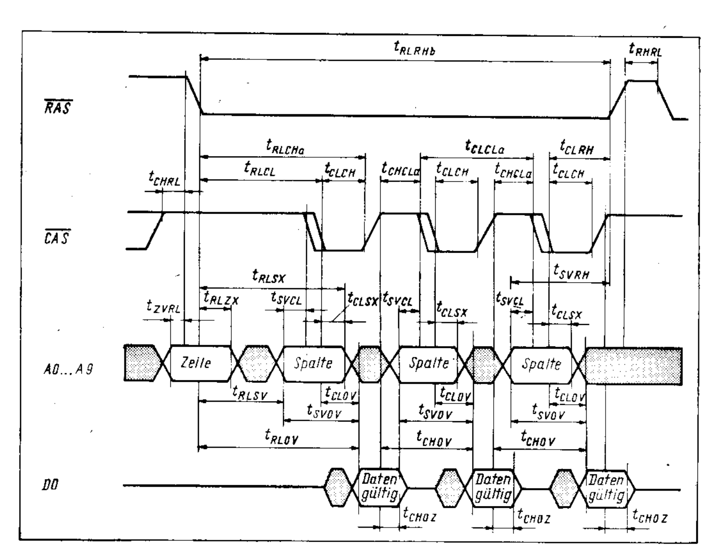


Bild 8 FAST PAGE MODE READ CYCLE (graue Flächen: Pegel nicht definiert)

gang hochohmig. Kommt die aktive Flanke von $\overline{WR}$ jedoch nach der aktiven $\overline{CAS}$-Flanke, aber vor Ablauf der Zugriffszeit, kann weder von einem Early-Write- noch von einem Read-Write-Zyklus sicher gesprochen werden, weil der Ausgang kurzzeitig unbestimmt ist und damit keine gültigen Lesedaten garantiert sind. Da jedoch das einwandfreie Schreiben gesichert ist, wird diese Betriebsart auch Late-Write-Zyklus genannt.

Fast Page Mode

Der *Fast Page Mode* erlaubt eine höhere Datenrate als der Normalbetrieb. Bei dieser Betriebsart erfolgen die Zugriffe in einem bestimmten Bereich des Speichers, nämlich einer Seite (page) von 1024 Bit. Es handelt sich dabei um eine Zeile der Speichermatrix, die durch die Zeilenadresse definiert ist. Der Vorteil des Fast Page Mode liegt darin, daß bei aufeinanderfolgenden Zugriffen innerhalb einer Seite die Zeilenadresse nur zu Beginn einmal übertragen werden muß und für jeden weiteren Zugriff lediglich die Spaltenadresse erforderlich ist. Bild 8 zeigt die Lesezyklen im Fast Page Mode. Schreibzyklen und Read-Modify-Write-Zyklen laufen entsprechend ab. Der Fast Page Mode wird dadurch ausgelöst, daß mit $\overline{RAS}$ auf Low die Zeilenadresse gehalten wird, während mit jeder aktiven $\overline{CAS}$-Flanke unter der angelegten Spaltenadresse ein Zyklus abläuft. Bei Lesezyklen gelten ähnliche Zusammenhänge bezüglich Zugriffszeit, wie im Abschnitt Lesezyklus beschrieben.

Im Fast Page Mode ist die dynamische Verlustleistung geringer, weil der interne Zugriff auf die ganze Zeile der Matrix nur zu Beginn erforderlich ist, während jeder weitere Zugriff nur noch eins der vorselektierten Bits betrifft. Zu beachten ist allerdings, daß eine Seite nicht beliebig lange aktiviert sein kann. Zum einen ist aufgrund interner dynamischer Vorgänge (floatende Wortleitung kann nicht beliebig lange überhöhtes Potential führen) die $\overline{RAS}$-Pulsbreite auf 100 µs begrenzt, zum anderen sei an den regelmäßig erforderlichen Refresh erinnert.

Im Unterschied zum Fast Page Mode wird im *Static Column Mode* ein Speicherzyklus innerhalb der „Seite" durch den Spaltenadreßwechsel ausgelöst, ohne daß eine aktive $\overline{CAS}$-Flanke erforderlich ist. Damit arbeitet der Speicher in dieser Betriebsart wie ein vollstatischer, ungetakteter Speicher. Aufgrund der nicht mehr erforderlichen Synchronisation mit $\overline{CAS}$ und dem damit verbundenen Wegfall einer $\overline{CAS}$-Vorladezeit verringert sich die Zykluszeit beim Static Column Mode.

Wegen der großen Ähnlichkeit von Static Column Mode und Fast Page Mode ist es möglich, beim U 61000-Chip durch Änderung einer Maskenebene beide Betriebsarten von einem Grundentwurf abzuleiten.

Verlustleistung

Ein nicht unwesentlicher Aspekt bei der Auswahl von Speicherschaltkreisen ist deren Verlustleistung. Dies ist nicht im Sinne des Energiesparens zu verstehen, sondern weil durch geringe Leistungsaufnahme des Speichersystems der Aufwand bei der Spannungsversorgung (Netzteile) sinkt und eine netzunabhängige Versorgung, zum Beispiel mit Notlaufbatterien vereinfacht oder überhaupt erst möglich wird.

Die Verlustleistung von dynamischen Speichern wird im wesentlichen von drei Anteilen im Stromverbrauch, dem Ruhestrom, dem Betriebsstrom und dem Refresh-Strom, bestimmt.

Der **Ruhestrom**, der aufgrund der verwendeten CMOS-Technik sehr niedrig liegt, ist immer aufzubringen und dient dem Ausgleich von internen Leckströmen und dem Erzeugen interner Referenzspannungen. Durch Anlegen von CMOS-Pegeln, besonders an $\overline{RAS}$ und $\overline{CAS}$, kann die Ruhestromaufnahme deutlich gegenüber der Verwendung von TTL-Pegeln gesenkt werden, was in zwei verschiedenen Werten für den Ruhestrom spezifiziert ist. ($I_{CCR1} = 2$ mA, $I_{CCR2} = 1$ mA)

Der **Betriebsstrom** wird hauptsächlich durch die Stromimpulse bei internen Umladevorgängen hervorgerufen, die immer dann auftreten, wenn Pegeländerungen an den Strobe-Eingängen $\overline{RAS}$ und $\overline{CAS}$ auftreten. Die spezifizierte Stromaufnahme (I_{CC0} = 50 mA) gilt für die minimale Zykluszeit und sinkt bei langsamerem Betrieb des Speichers im Verhältnis von minimaler zu tatsächlicher Zykluszeit. Liegen die Strobe-Signale längere Zeit auf Low-Potential, spricht man auch von einer aktiven statischen Stromaufnahme, die aber stets kleiner als die dynamische Stromaufnahme ist. Im Falle des Fast Page Mode ist eine geringere Betriebsstromaufnahme spezifiziert ($I_{CC2} = 30$ mA), da nur noch die durch $\overline{CAS}$ ausgelösten Umladevorgänge ablaufen, die die Matrix mit ihren großen Kapazitäten nicht berühren.

Der **Refresh-Strom** ist etwas geringer als der Betriebsstrom, da die Ausgangstreiber und weitere periphere Schaltungen beim Auffrischen nicht aktiviert werden. Der spezifizierte Wert ($I_{CC1} = 50$ mA) unterscheidet sich allerdings nicht von I_{CC0}. Innerhalb der Refreshzeit $t_{Ref} = 8$ ms sind 512 Refreshzyklen erforderlich, womit der Mittelwert dieses Stromanteils folgendermaßen ermittelt wird:

$$I_{CCRef} = I_{CC1} \times \frac{t_{RLRL}}{t_{Ref}} \times 512$$

Zusätzlich zu diesen drei Stromanteilen ist der Ausgangslaststrom an DO prinzipiell zu berücksichtigen. Meist kann er allerdings vernachlässigt werden, weil I_{CC0} oft wesentlich größer als der Ausgangslaststrom ist.

Die Verlustleistung ergibt sich aus der Summe dieser Stromanteile, multipliziert mit der maximalen Versorgungsspannung, wobei es sinnvoll ist, diesen Wert für eine konkrete Speicherkonfiguration (Leiterplatte o. ä.) zu berechnen. Damit ergibt sich:

$$P_{Vmax} = (I_{CC0} \times j + I_{CCR} \times k + I_{CCRef} \times i) \times U_{CCmax}$$

mit

i = Gesamtzahl der Speicherschaltkreise
j = Anzahl der aktiven Schaltkreise
k = i – j = Anzahl der inaktiven Schaltkreise

Im folgenden Beispiel soll die Verlustleistung einer Speicherkonfiguration berechnet werden. Eine Speicherplatine mit einer Organisation von 4 M × 16 Bit soll bei minimaler Zykluszeit betrieben werden. Damit ergibt sich:

i = 64 Schaltkreise U 61000
j = 16 Schaltkreise U 61000
k = 48 Schaltkreise U 61000

Mit den spezifizierten Werten des U 61000 DC12:

$I_{CC0} = 50$ mA (Normalbetrieb)
$I_{CCR1} = 2$ mA (Ansteuerung mit TTL-Pegeln)
$I_{CC1} = 50$ mA
$t_{RLRL} = 220$ ns
$t_{Ref} = 8$ ms

erhalten wir drei Stromanteile

$I_{CC0} \times j = 800$ mA
$I_{CCR} \times k = 96$ mA

$$I_{CCRef} \times i = \frac{50 \times 512 \times 220 \times 64}{8 \times 10^6} = 45\,\text{mA}$$

Als maximale Gesamtverlustleistung für die gewählte Speicherkonfiguration ergibt sich damit:

$P_{Vmax} = (800 + 96 + 45) \times 5{,}5\,\text{mW} = \mathbf{5{,}18\,W}$

Betriebsbedingungen
Betriebsspannung, Temperatur

Die Betriebsspannung für 1-MBit-DRAMs liegt generell bei $U_{CC} = 5\,V \pm 10\,\%$ im Temperaturbereich von 0 bis 70 Grad Celsius. Jedoch ist kein Speicher so dimensioniert, daß diese Werte nur knapp eingehalten werden, weil dann Fertigungstoleranzen sofort zu fehlerhaften Bauelementen führen würden. Zielstellung ist es, daß die Schaltkreise bis zu Betriebsspannungsabweichungen von über 20 % arbeiten. Hauptproblem des Speichers sind hohe Temperaturen, da die Selbstentladung der Speicherkondensatoren exponentiell mit der Temperatur ansteigt, was zu einer Unterschreitung der spezifizierten Refreshzeit und einem damit verbundenen Datenverlust führt. Neben der Verlustleistung des Speichersystems sind auch die Flankensteilheiten und Spitzenwerte der Stromimpulse bei der Auslegung der Spannungsversorgung wichtig. Um zu verhindern, daß die Betriebsspannung bei Stromspitzen zu stark zusammenbricht, sind schaltkreisinterne Maßnahmen durch entsprechende Stützkondensatoren auf der Leiterplatte zu ergänzen.

In diesem Zusammenhang sei noch das Verhalten beim Anlegen der Versorgungsspannung erwähnt. Eine interne Logik sorgt für ein gezieltes, stufenweises Zuschalten der Funktionsgruppen des Speichers. Damit wird ein *Latch-up* beim Einschalten wirkungsvoll verhindert, wenn ein ordnungsgemäßer Betriebsspannungsanstieg von kleiner als 0,1 V je ns gewährleistet ist. Eine weitere Forderung beim Einschalten ist, nach dem Erreichen von U_{CC} 200 µs zu warten und anschließend 8 $\overline{RAS}$-Zyklen zu durchlaufen. Mit dieser *wake-up-procedure* wird gesichert, daß alle internen Pegel auf dem richtigen Wert liegen und der Speicher voll funktionsfähig ist.

Ein- und Ausgangspegel

Die Ein- und Ausgänge des U 61000 sind TTL-kompatibel. CMOS-Schaltkreise lassen sich leichter ansteuern als TTL-Schaltkreise, da ihre Pegel in der Nähe der Betriebsspannungspotentiale U_{CC} und U_{SS} liegen. Neben der höheren Störsicherheit ist damit auch eine geringere Ruheverlustleistung verbunden.

Flankensteilheiten der Steuersignale von 5 ns und geringer sichern die spezifizierte Arbeitsgeschwindigkeit. Damit ist allerdings ein Überschwingen der Eingangssignale verbunden. DRAM-spezifisch ist ein negatives Überschwingen von –1 V und ein positives Überschwingen auf 6,5 V zulässig. In diesem Zusammenhang sei auch noch erwähnt, daß es in schnellen oder auch räumlich ausgedehnten Speichersystemen zu störenden Reflexionen auf den Daten- und den Taktleitungen kommen kann.

Soft-Errors

Ein wichtiges Problem bei dynamischen Speichern sind die sogenannten Soft-Errors. Diese sind einmalige Fehler, die durch Ein-

wirkung von Alphateilchen auf den Speicherkondensator, die Bitleitung oder den Leseverstärker verursacht werden. Diese Teilchen entstehen beim radioaktiven Zerfall von Uran, Thorium und weiteren Elementen, die in Spuren in jedem Gehäusematerial vorkommen.
Beim Durchgang von Alphateilchen durch die genannten Gebiete werden Ladungsträgerpaare erzeugt, und die im p-Substrat als Minoritätsladungsträger entstehende Elektronenwolke kann in die n-leitenden Speichergebiete gelangen und die gespeicherte Information vernichten. Abgesehen von der falschen Information wird der Speicher dabei nicht beschädigt. Gemessen werden die Soft-Errors in der Einheit *fit*, die angibt, wieviel Fehler in 10^9 Betriebsstunden auftreten.
Durch die Verwendung strahlungsarmer Gehäusematerialien und durch Implantation einer Potentialbarriere unterhalb der Speicherkondensatoren und Bitleitungen, die eine Elektronenwolke unwirksam macht, ist beim U 61000 eine geringe Soft-Error-Empfindlichkeit erreicht worden.

✉ KONTAKT ☏
VEB Forschungszentrum Mikroelektronik Dresden, PSF 34, Grenzstraße 28, Dresden, 8080; Tel. 58 84 20

8-Bit-CMOS-Mikroprozessorsystem U 84C00

Michael Ritter
VEB Mikroelektronik „Karl-Marx“ Erfurt, Forschungszentrum

Michael Ritter – Jahrgang 1961 – studierte von 1982 bis 1987 an der Technischen Hochschule Ilmenau in der Fachrichtung Informationstechnik und theoretische Elektrotechnik. Seit 1987 ist er als Applikationsingenieur in der Abteilung Mikroprozessorapplikation im Forschungszentrum des VEB Mikroelektronik „Karl Marx“ Erfurt tätig.

Vorbemerkungen

Eine Betrachtung des internationalen 8-Bit-Mikroprozessor-Marktes zeigt den Trend, daß in den letzten Jahren eine große Anzahl von Systemen in CMOS-Technologie eingeführt wurde. Dabei wurden einige grundsätzliche Gemeinsamkeiten deutlich. Erstens werden immer ganze Familien, das heißt Prozessoren und zugehörige Peripherieschaltkreise, in der neuen Technologie gefertigt. Zweitens versuchen alle Hersteller, CMOS-Prozessoren anzubieten, die kompatibel zu bestehenden NMOS-Lösungen sind. Drittens ist eine erkennbare Gemeinsamkeit, daß Weiterentwicklungen der klassischen 8-Bit-Prozessoren mit verbesserten Architekturen angeboten werden. So entstanden zum Beispiel der 80C85 (Intel) aus dem 8080 und der HD64180 (Hitachi) aus dem Z 80. Vom international dominierenden 8-Bit-Prozessorsystem Z 80 existieren diverse Weiterentwicklungen in CMOS-Technologie. Unter anderem bieten Zilog, Toshiba, Sharp und Hitachi solche Systeme an.
Im VEB Mikroelektronik „Karl-Marx“ Erfurt (MME) wird die Familie U 84C00 in CMOS-Technologie als Weiterentwicklung der Familie U 880 gefertigt. Diese Schaltkreise eignen sich vor allem zum Einsatz in mobilen Meßgeräten und Handheld-Computern. Sie können aber auch in bestehenden Lösungen die Schaltkreisfamilie U 880 ablösen.

Die Schaltkreisfamilie U 84C00 Vorstellung

Die Schaltkreisfamilie U 84C00 ist ein leistungsfähiges 8-Bit-Mikroprozessorsystem in CMOS-Technologie, welches neben der CPU eine Reihe von Peripheriebauelementen umfaßt. Die Schaltkreise werden in den Frequenzklassen 2,5 MHz und 4,0 MHz angeboten. Zu einigen Typen gibt es Anfallbauelemente mit der Bezeichnung U 84Cxx DC02-1. Diese unterscheiden sich von den Standardtypen dadurch, daß sie nicht in den Schlafzustand übergehen können. Alle anderen Parameter entsprechen denen der Standardbauelemente. Tafel 1 zeigt eine Übersicht über die vom MME angebotenen Typen.

Vergleich der Familien U 84C00 und U 880

Die Schaltkreise der Familie U 84C00 sind grundsätzlich kompatibel zu den Schaltkreisen der Familie U 880. Das gilt für alle Befehle ebenso wie für alle Hardwareparameter einschließlich der TTL-Inkompatibilität des Takteinganges (siehe auch Bilder 1 und 2). Da die Schaltkreisfamilie in CMOS-Technologie gefertigt wird, besitzt sie gegenüber Schaltkreisen der U 880-Familie den gewichtigen Vorteil des geringen Stromverbrauchs. Für den Anwender sind damit Applikationslösungen möglich, die sich durch geringe Eigenerwärmung, kleinere Netzteile und kleinere Gehäuse auszeichnen. Außerdem ist der Einsatz von Batterien zur Stromversorgung mit vertretbarem Aufwand möglich.
Tafel 2 zeigt einen Vergleich der Stromaufnahme der Schaltkreise der U 84C00-Familie mit der der U 880-Familie.
Ein CMOS-Bauelement besteht aus Grundbausteinen, die aus komplementären Feldeffekttransistoren aufgebaut sind. Im statischen Betrieb ist immer nur einer der beiden Transistoren leitend. Die Verlustleistung wird im wesentlichen durch Leckströme bestimmt. Während eines Umschaltvorganges müssen die Gatekapazitäten umgeladen werden, und beide Transistoren sind kurzzeitig leitend. Daraus folgt, daß die Verlustleistung von

Tafel 1 Übersicht über die vom MME angebotenen Typen

Typ	2,5 MHz Standardtyp	2,5 MHz Anfalltyp	4,0 MHz Standardtyp
CPU	U 84C00 DC02	U 84C00 DC02–1	U 84C00 DC04
PIO	U 84C20 DC02	U 84C20 DC02–1	U 84C20 DC04
CTC	U 84C30 DC02	U 84C30 DC02–1	U 84C30 DC04
SIO	U 84C40 DC02	–	U 84C40 DC04
DART	U 84C70 DC02	U 84C70 DC02–1	U 84C70 DC04
CGC	U 84C97 DC02	–	U 84C97 DC04

CPU – zentrale Verarbeitungseinheit
PIO – parallele Ein-/Ausgabeeinheit
CTC – Zähler-/Zeitgebereinheit
SIO – serielle Ein-/Ausgabeeinheit
DART – asynchrone serielle Ein-/Ausgabeeinheit
CGC – Taktgenerator/-controller

Tafel 2 Vergleich der Stromaufnahme (4 MHz)

Typ	U 84C00-Familie	U 880-Familie
CPU	25 mA	200 mA
PIO	5 mA	100 mA
SIO	15 mA	130 mA
CTC	7 mA	120 mA
CGC	4 mA	–

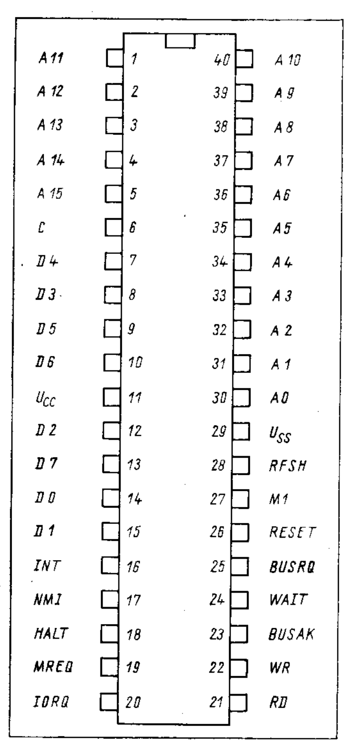


Bild 1 Pinbelegung des U 84C00

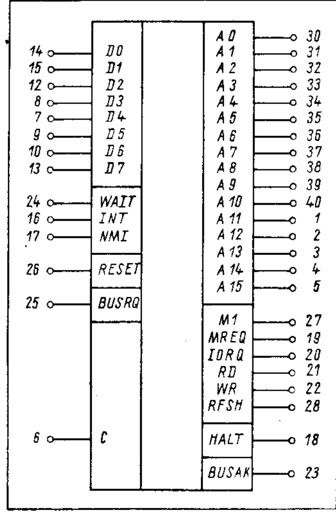


Bild 2 Schaltbild des U 84C00

CMOS-Mikroprozessorschaltkreisen im statischen Betrieb außerordentlich klein ist und mit steigender Frequenz zunimmt.
Die Schaltkreise der Familie U 84C00 sind im Unterschied zu den Schaltkreisen der Familie U 880 intern statisch aufgebaut. Daraus folgt, daß es keine untere Grenzfrequenz für diese Schaltkreise gibt. Werden sie mit der Taktfrequenz Null betrieben (Anhalten des Systemtaktes) bleiben alle internen Informationen erhalten. Bei Wiederzuschalten des Taktes arbeitet das System an der Stelle weiter, an der die Bearbeitung zuvor unterbrochen wurde. Diese Tatsache wird beim sogenannten Schlafzustand (power down mode) ausgenutzt. Diese Betriebsart bietet zusätzlich zu der Eigenschaft einer sehr günstigen Leistungsbilanz im Betriebsfall die Möglichkeit, den Stromverbrauch der Schaltkreise noch weiter zu minimieren. Wenn keine Aktivität des Rechners erforderlich ist, können die Schaltkreise in den Schlafzustand versetzt werden. Dazu wird nach Ausführung des Befehls HALT in einem festgelegten Regime der Systemtakt angehalten. Die ordnungsgemäße Einnahme des Schlafzustandes wird vom Taktgenerator/-controller (CGC) gesteuert. Die Stromaufnahme eines jeden Schaltkreises im Schlafzustand ist kleiner als 10 Mikroampere.

Einnahme des Schlafzustandes durch die CPU

Der Systemakt kann zu jedem Zeitpunkt im Betriebsfall angehalten werden. Daß der Betriebsstrom der CPU im Ruhezustand dem Standby-Strom von < 10 Mikroampere entspricht, wird allerdings nur dann garantiert, wenn das Anhalten des Taktes im genau richtigen zeitlichen Ablauf erfolgt. Zum ordnungsgemäßen Übergang in den Schlafzustand muß der Takt auf dem Low-Pegel des 4. Taktzyklus des Maschinenzyklus, der auf den Befehl HALT folgt, gehalten werden. Bild 3 zeigt den vom CGC gesteuerten ordnungsgemäßen Übergang in den Schlafzustand.

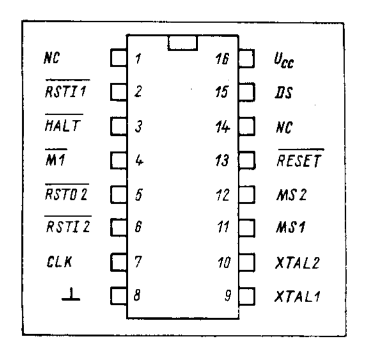


Bild 3 Pinbelegung des CGC U 84C97

Durch Anlegen des Systemtaktes an die CPU kehrt diese wieder aus dem Schlafzustand zurück. Auch dieser Übergang wird vom CGC gesteuert. Der HALT-Zustand wird auf bekannte Weise entweder durch RESET, INT oder NMI aufgehoben.

Beschreibung des U 84C97

Allgemeines

Der Schaltkreis U 84C97 ist ein Taktgenerator/-controller für die CPU U 84C00 und die dazugehörigen peripheren Schaltkreise. Er wird in CMOS-Technologie hergestellt und befindet sich in einem 16poligen DIL-Gehäuse (Bilder 3 und 4).
Der CGC hat in einer Applikation der Familie U 84C00 zwei Aufgaben zu erfüllen:

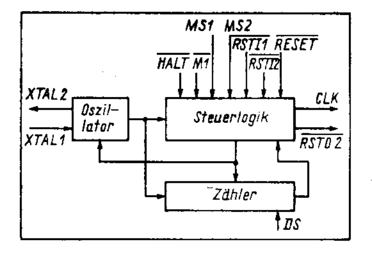


Bild 4 Schaltbild des U 84C97

Tafel 3 Pinbelegung des Schaltkreises U 84C97

Pinnummer	Bezeichnung	Beschreibung
1	NC	nicht belegt
2	RSTI1	Eingang zum Wiederzuschalten des Taktes, pegelgetriggert, üblich für INT-Anforderung
3	HALT	Eingang für HALT-Signal der CPU
4	M1	Eingang für M1-Signal der CPU
5	RSTO2	zur RSTI2 korrespondierender Ausgang, üblicherweise mit NMI der CPU verbunden
6	RSTI2	gelatchter Eingang zum Wiederzuschalten des Taktes, flankengetriggert, üblich für NMI-Anforderungen
7	CLK	Taktausgang
8	U_{SS}	Masse
9	XTAL1	Quarzanschluß (X_{in})
10	XTAL2	Quarzanschluß (X_{out})
11	MS1	Eingang für Mode Select
12	MS2	Eingang für Mode Select
13	RESET	Eingang zum Wiederzuschalten des Taktes, üblich für RESET-Signal
14	NC	nicht belegt
15	DS	Eingang zur Auswahl der Aufwärmzeit im STOP-Modus
16	U_{CC}	Betriebsspannung

- Bereitstellung des Systemtaktes
- Organisation des Übergangs in den Schlafzustand.

In Tafel 3 werden die Anschlüsse des Schaltkreises erläutert.

Betriebsarten des Schaltkreises U 84C97

Der Schaltkreis U 84C97 kann in drei Betriebsarten betrieben werden: RUN MODE, IDLE MODE, STOP MODE.
Die Auswahl der Betriebsarten erfolgt über die Anschlüsse MS1 und MS2. Dabei gilt die Zuordnung nach Tafel 4.

Tafel 4 Betriebsarten des Schaltkreises U 84C97

MS1	MS2	MODE	Beschreibung
1	1	RUN	Systemtakt wird immer zur Verfügung gestellt.
0	x	IDLE	Systemtakt wird angehalten, aber interner Oszillator läuft weiter.
1	0	STOP	Alle internen Operationen werden gestoppt.

HALT-Operation in jedem Modus
x = beliebig

Die Betriebsart **RUN MODE** wird dann eingestellt, wenn die bestehende Schaltung nie in den Schlafzustand übergehen soll. Nach Erkennen des HALT-Befehls geht die CPU in den HALT-Zustand über. Der CGC liefert den Systemtakt weiter. Die CPU führt während des HALT-Zustandes NOP-Befehle aus. In dieser Betriebsart hat der Taktcontroller eine Stromaufnahme von etwa 4 mA. Bild 5 zeigt das Taktdiagramm für diese Betriebsart.
In Bild 6 ist das Timing für den Fall dargestellt, daß die CPU einen HALT-Befehl (76H) erkennt und der Taktcontroller auf die Betriebsart **IDLE MODE** eingestellt ist. Das Aktivwerden des HALT-Signals der CPU wird durch die fallende Taktflanke von T4 synchronisiert.

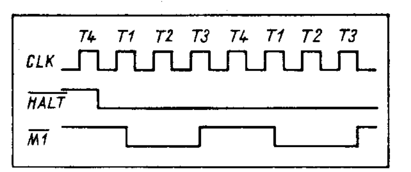


Bild 5 Taktdiagramm im RUN MODE

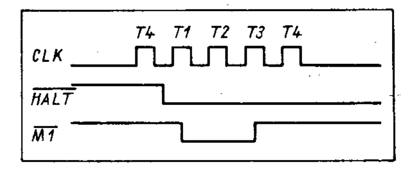


Bild 6 Taktdiagramm im IDLE/STOP MODE

Die steigende Flanke des Signals M1 bewirkt, daß der CGC den Taktausgang nach der fallenden Flanke von T4 auf Low hält. Das Anhalten des Taktes überführt die CPU in den Schlafzustand. Der interne Oszillator des CGC läuft in dieser Betriebsart weiter. Der Schaltkreis hat in diesem Zustand eine Stromaufnahme von zirka 500 Mikroampere.
Im **STOP MODE** wird nach dem HALT-Befehl der CPU die gleiche Operation wie im **IDLE** MODE ausgeführt. Der Unterschied besteht darin, daß der CGC im STOP MODE alle Aktionen einstellt. Dabei wird auch der interne Oszillator abgeschaltet. Im abgeschalteten Zustand hat der CGC in dieser Betriebsart eine Stromaufnahme von < 10 Mikroampere.

Das Wiederbereitstellen des Taktes

Es gibt drei Eingänge, die dazu benutzt werden können, den Schlafzustand zu beenden: RESET, RSTI1 und RSTI2. Bild 7 zeigt den Ablauf des Wiederbereitstellens des Taktes im **IDLE MODE**. Nachdem eine Anforderung erfolgte, wird nach geringer Verzögerung der Takt bereitgestellt.

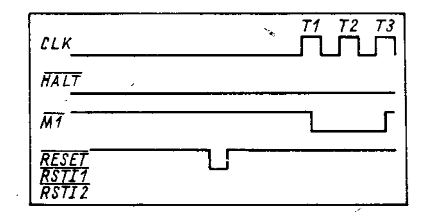


Bild 7 Wiederbereitstellen des Taktes im IDLE MODE

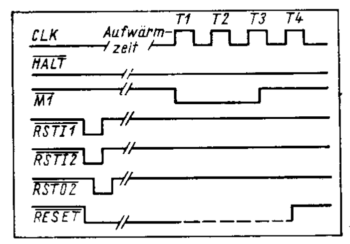


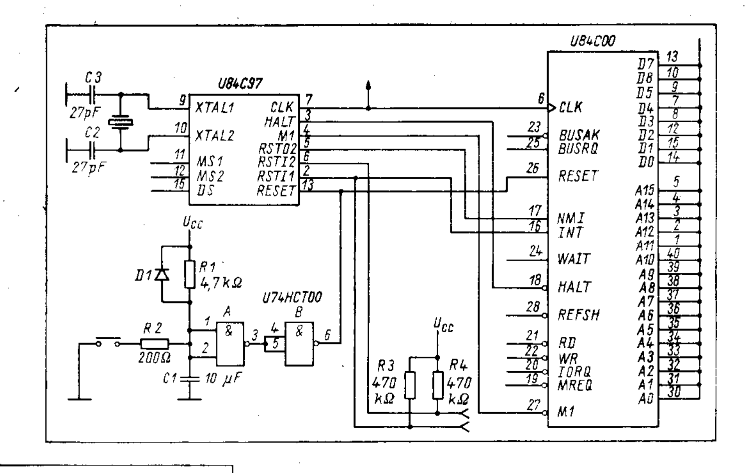
Bild 8 Wiederbereitstellen des Taktes im STOP MODE

Bild 8 zeigt den Ablauf des Wiederbereitstellens des Taktes im **STOP MODE**. Da im STOP MODE der interne Oszillator ausgeschaltet ist, wird eine Aufwärmzeit benötigt, um das ordnungsgemäße Anschwingen gewährleisten zu können. Es können zwei verschiedene Werte für die Anschwingzeit eingestellt werden. Dazu wird das Pin DS verwendet.

Liegt DS auf High, wird eine Aufwärmzeit von 2^{15} Taktperioden eingestellt. Liegt DS auf Low, wird eine Aufwärmzeit von 2^{18} Taktperioden eingestellt. Wird zum Neustart des CGC der RESET-Eingang verwendet, dann wird keine Aufwärmzeit generiert. Wenn die RESET-Eingänge von CPU und CGC verbunden sind, muß also von außen gewährleistet sein, daß das RESET-Signal hinreichend lange auf Low gehalten wird, um ein ordnungsgemäßes Rücksetzen der CPU sichern zu können.

Bild 9 zeigt die Standardbeschaltung von CPU und CGC. Es muß beim Einsatz des Systems konsequent darauf geachtet werden, daß keine Eingänge unbeschaltet bleiben. Auf Grund der hohen Eingangsimpedanzen der CMOS-Schaltkreise kommt es sonst zu Störungen im Betrieb. Der Abschluß kann mit hochohmigen Widerständen bis etwa 500 kOhm erfolgen.

Bild 9 Beschaltung von CPU und CGC



☒ KONTAKT ☏

VEB Mikroelektronik „Karl Marx" Erfurt, Forschungszentrum, Applikationszentrum Bauelemente, Rudolfstraße 47, Erfurt, 5010; Tel. 5 10 76, App. 58

Hybridschaltkreise aus Hermsdorf

Dr. Bernd Racurow
Kombinat VEB Keramische Werke Hermsdorf, Betrieb Mikroelektronik

Bernd Racurow studierte von 1959 bis 1965 Hochfrequenztechnik an der Technischen Universität Dresden. Dort promovierte er auch 1976 im Rahmen einer außerplanmäßigen Aspirantur zu Problemen der Sicherung der elektrischen Kennwerte von Hybridschaltkreisen. Seit 1965 beschäftigte er sich in den Keramischen Werken Hermsdorf mit keramischen Piezofiltern für Rundfunk- und Fernsehanwendungen. Im Betrieb Mikroelektronik des KKWH wirkt Bernd Racurow seit 1969 in verschiedenen Funktionen, seit 1986 ist er Fachdirektor Forschung und Entwicklung für Hybridtechnik. Seine Spezialgebiete sind die Schaltungs- und die Meßtechnik von Hybridschaltkreisen sowie die Digital-Analog-Wandler.

Das Kombinat VEB Keramische Werke Hermsdorf ist der zentrale Hersteller für Hybridschaltkreise in der DDR. Damit steht vor ihm die Aufgabe, vor allem multivalent einsetzbare Hybridschaltkreise zu fertigen und kundenspezifische Anforderungen mit hochproduktiven Technologien zu erfüllen. In Abweichung vom internationalen Trend wurde in den Keramischen Werken Hermsdorf der Dünnfilmtechnik gegenüber der Dickschichttechnik der Vorzug gegeben. Es wurde der Nachweis erbracht, daß die Massenproduktion von hochgenauen Widerstandsnetzwerken mit einer Widerstandstoleranz von 0,1 % in einem Wertespektrum von 0,1 KΩ bis 1 MΩ und einer Relativtoleranz von 0,01 %, bei einem TK_R von ≥ 10 ppm/K und einem Relativ-TK_R von 1 ppm/K mit hoher Effektivität möglich ist. Die Strukturierung und der Widerstandsabgleich erfolgen mit dem Elektronenstrahl mit höchster Produktivität. Für ein Widerstandsnetzwerk auf Glas mit 10 Widerständen werden 2,5 s benötigt. Zur Zeit werden pro Jahr 8 Mio Stück Widerstandsnetzwerke in Dünnfilmtechnik auf Glas in Bedampfungstechnologie produziert.

Es war nur konsequent, daß diese Spitzenstellung in der Dünnfilmtechnik die Entwicklung der Hybridschaltkreise mit hohen Anforderungen an die Genauigkeit und an die Stabilität der elektrischen Parameter förderte.

1981 wurde mit der Fertigung eines Digital-Analog-Wandler-Systems mit einer Auflösung von 15 Bit und einer Linearität von 0,1 % zum Einsatz in CNC-Steuerungen begonnen. Durch eine neuartige Schaltungstechnik der Stromquelle wurde eine außergewöhnliche Stabilität der Schaltung erreicht, so daß eine Linearität von 0,1 ohne Funktionsabgleich, nur durch Montage von vorabgeglichenen Widerstandschips auf Glas mit den Halbleiterchips auf einer Dickschicht-Grundplatte gewährleistet wurde.

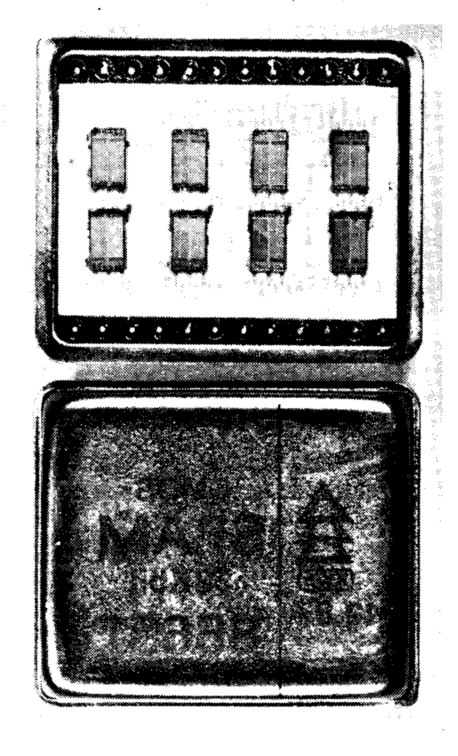


Bild 1 Statischer 32-KBit-CMOS-RAM (unverkappt und verkappt)

1983 folgte ein 12-Bit-DA-Wandler für Standardanwendungen nach dem gleichen Prinzip. Der Typ DAC 32 erfordert nur einen geringfügigen Funktionsabgleich, der in durchschnittlich 3 Minuten erfolgt. 1984 wurde ein Instrumentationsverstärker entwickelt, der mit Chips des MAA 725 (µA 725) höchste Anforderungen an die Temperaturabhängigkeit der elektrischen Parameter erfüllt. Im gleichen Jahr wurden zur Abrundung dieses Sortiments von Präzisionsschaltkreisen der analogen Peripherie von Mikrorechnern ein 12-Bit-AD-Wandler und ein Trennverstärker der Genauigkeitsklasse 0,1 % für eine Isolationsspannung von 1 kV nach dem Prinzip der *Flyback*-Modulation mit gedrucktem Übertrager in das Fertigungsprogramm aufgenommen.

Seit 1980 wurden die Grenzen der Dünnfilmtechnik auf Glas mit aufgedampften Schichten immer deutlicher. Glas ist ein relativ schlechter Wärmeleiter, so daß die weitere Erhöhung des Integrationsgrades der Widerstandsnetzwerke und der Applikationen mit höherem Leistungsumsatz in Dünnschichttechnik den Einsatz neuer Substratmaterialien, wie passiviertes Silizium und Keramik, erforderten. Ab 1984 wurde daher die Hochrate-Sputtertechnologie eingeführt. Das Institut „Manfred von Ardenne", mit dem unser Betrieb seit vielen Jahren durch eine erfolgreiche Zusammenarbeit verbunden ist, stellte die rechnergesteuerten Anlagen dafür zur Verfügung. Widerstandschips auf passiviertem Silizium werden mit einem Temperaturkoeffizienten von ≦ 5 ppm/K und einem Relativ-TK von ≦ 0,2 ppm/K in DA-Wandler mit einer Genauigkeit von bis zu 16 Bit eingesetzt. 1986 wurde der DAC 4071, ein zum international bekannten DAC 71 kompatibler Typ mit einer Auflösung von 16 Bit und einer Linearität von 14 Bit in unser Fertigungsprogramm aufgenommen. Zur Zeit befindet sich ein echter 16-Bit-Wandler in Entwicklung. Durch Nutzung des Prinzips der digitalen Korrektur läßt sich mit Hilfe eines zusätzlichen Speichers, in dem die Korrekturwerte digital gespeichert werden, jederzeit die 16-Bit-Linearität abgleichen. Die Sputtertechnik auf unglasierter und glasierter Dickschichtkeramik eröffnet völlig

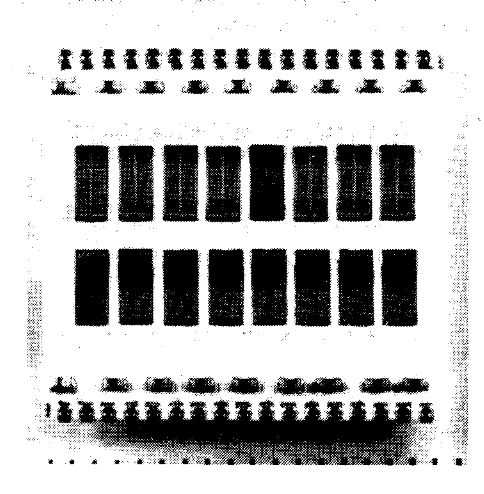


Bild 2 Dynamischer 1-MBit-RAM (unverkappt)

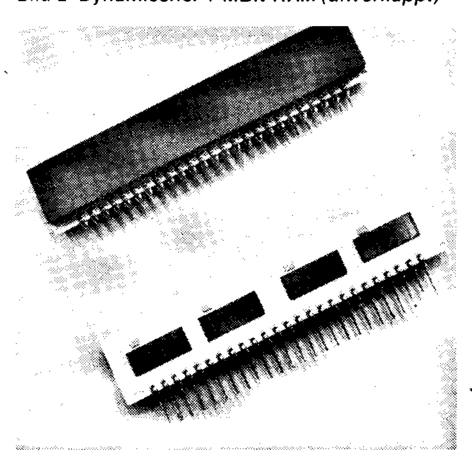


Bild 3 Dynamischer 4-MBit-RAM (verkappt und unverkappt)

neue applikative Möglichkeiten. Es können Widerstandsnetzwerke, die bisher nur mit Edelmetallpasten in Dickschichttechnik möglich waren, hergestellt werden, und auf glasierter Keramik werden Netzwerke für die Thermodrucktechnik produziert. In dieser Technologie sind feinste Strukturen mit 8 Bildpunkten pro mm das Ziel der Entwicklung von Netzwerken für Thermodruckköpfe. Es sei nur erwähnt, daß auch die notwendigen Abrieb-Schutzschichten (SiO_2 und $CrSiO_x$), die einen vorzeitigen Verschleiß der Köpfe verhindern, gesputtert werden.
Selbstverständlich gibt es auch unter den Bedingungen einer gegenüber dem internationalen Stand produktiveren Dünnschichttechnologie Anwendungen, die sinnvoll nur in Dickschichttechnik oder in Kombinationstechnik, daß heißt mit Dünnschicht-R-Chips, gefertigt werden können. Während international die prozentualen Anteile von Dickschicht- zu Dünnschichtanwendungen wertmäßig etwa 80/20 betragen, weichen wir mit 60/40 davon ab.
Auf der Grundlage einer dem internationalen Stand entsprechenden Dickschichttechnologie wurden Erzeugnisse entwickelt, die in besonderer Weise die Vorteile der Dickschichttechnik, wie Bewältigung hoher Verlustleistungen, Hochohmigkeit, Mehrebenentechnik, um nur einige zu nennen, nutzen. Gegenüber der SMD-Technik (surface mounted device – oberflächenmontierbares Bauelement) auf Leiterplatten ist auch die Möglichkeit des Funktionsabgleichs (activ trimming) ein wesentlicher Vorteil. Schwerpunkterzeugnisse in Dickschichttechnik werden in erster Linie mit SMD-gerechten Bauelementen, vor allem Transistoren in SOT 23- und SOT 89-Gehäusen, sowie mit ICs in SO-Gehäusen und in Chipcarriern hergestellt.
Besonders profiliert in der Anwendung der Dickschichttechnologie hat sich unser Betrieb auf den Gebieten Medizinelektronik und KFZ-Technik.
Nach Hybridschaltkreisen für Herzschrittmacher der ersten und zweiten Generation erfolgte 1988 die Produktionsaufnahme von zwei Hybridschaltkreisen (Ein- und Zweikammervariante) der dritten Generation. In den 90er Jahren wird es Hybridschaltkreise der 4. Generation geben, die eine Anpassung der Herzfrequenz über Sensoren ermöglichen.
In günstiger Weise lassen sich Hybridschaltkreise für Drehstromlichtmaschinen in Dickschichttechnologie herstellen. Durch Funktionsabgleich können die Sollspannungen 14 V bzw. 28 V eingestellt werden. In die Serie überführt wurden Steuerteile für elektronische 4-Takt-Zündungen.
Die Produktivität und Zuverlässigkeit der mikroelektronischen Technologien und natürlich auch der Hybridtechnik werden wesentlich durch automatisierte Montageverfahren bestimmt. In Zusammenarbeit mit dem VEB Elektromat Dresden (Betrieb im Kombinat Carl Zeiss JENA) wurde der vollautomatische Drahtbonder VADB 60 entwickelt. Er ermöglicht im *Teach-in*-Verfahren die vollautomatische Montage nicht nur eines Chips, sondern bei Einhaltung von vorgegebenen Bestückungstoleranzen das Bonden aller Chips in einem Hybridschaltkreis. Es können maximal 400 Bondbrücken in einem Hybridschaltkreis mit einer durchschnittlichen Produktivität von 3000 Bondbrücken pro Stunde kontaktiert werden.
Es liegt nahe, die automatisierten Montagetechnologien für SMD-Bauelemente und für Chips zur Fertigung von hochintegrierten Verdrahtungsträgern einzusetzen, die als Dickschichtmehrebenenschaltungen und Dünnschichteinebenenschaltungen hoher Strukturierungsdichte ($< 50\,\mu m$) zur Verfügung stehen. Der Übergang zu Verdrahtungsträgern auf der Basis von Dickschicht-Kupferpasten und zur Dünnschichtmehrebenentechnik entspricht internationalen Entwicklungen. Eine Applikation der Verdrahtungsträger sind hybride Speicher höchster Integration. 1985 wurden statische CMOS-RAMs mit einer Speicherkapazität von 32 KBit in die Fertigung überführt (siehe Bild 1). 1989 werden mehrere tausend dynamische 1-MBit-Speicher auf der Grundlage von 64-KBit-Chips produziert (Bild 2). Auf der Leipziger Frühjahrsmesse 1989 wurden Muster von 4-MBit-Speichern auf Hybridschaltkreisträgern vorgestellt, die aus sechzehn 256-KBit-Chips bzw. vier 1-MBit-Chips (Bild 3) bestehen. Sie demonstrieren die Möglichkeiten der Hybridtechnik hinsichtlich höchster Bauelementedichte und Integration.

Kontamination oder Reicht das Staubwischen in der Halbleiterfertigung?

Dr. Johannes Rolf Hillig
VEB Kombinat Mikroelektronik Erfurt

Mit dem Einsatz von Mikrorechnern für immer kompliziertere Aufgaben, zum Beispiel Bildverarbeitung oder Echtzeitsteuerungen, wächst die Forderung nach schnelleren Prozessoren und größeren Speichern. Das bewirkt bei den integrierten Schaltkreisen eine ständige Erhöhung der Transistorfunktionseinheiten pro Chip. Damit aber nicht auch die Chipflächen unendlich wachsen, müssen die Strukturgrößen der Transistoren verringert werden. Reichten für den 256-KBit-RAM noch Strukturen um 1,5 µm, so sind es beim 1-MBit-Chip schon 1,0 µm, und für den 16-MBit-Chip werden sogar Strukturen von 0,5 µm nötig sein. Mit den abnehmenden Strukturgrößen wirken sich bei der Chipherstellung Schmutzpartikel in steigendem Maße auf die Ausfallrate aus. Deshalb wird es immer komplizierter, einer Partikelablagerung auf den Siliziumscheiben, genannt Kontamination, entgegenzuwirken.

Kontamination im Halbleiterfertigungsprozeß

Nach der bei Mitsubishi /1/ üblichen Definition ist „Kontamination eine fremde Substanz, die einen determinierten Einfluß auf das Erzeugnis oder den Prozeß (die Technologie)" hat. Mit der Entwicklung der Nukleartechnik um 1940 wurden erstmals Forderungen an die Umgebungsluft gestellt, was zu speziellen Luftfiltern führte. Daraus entwickelten sich in den folgenden Jahren über Reinstarbeitsbänke (Laminar-Flow-Benches) die heute allgemein genutzten Rein- und Reinsträume. Diese Vorkehrungen dienten der Reduzierung „fremder Substanzen" in der Luft, das heißt der Kontamination aus der Luft. Mit der industriellen Nutzung des Halbleitereffekts in Form von Transistoren und Integrierten Schaltkreisen wurden die Anforderungen an die Kontaminationsreduzierung immer größer. So wurde beispielsweise in den 40er Jahren die „Fremd"- oder Kontaminationssubstanz noch in Prozent (1 : 100) oder Promille (1 : 1000) gemessen, während beispielsweise gegenwärtig Forderungen nach Reinheiten im ppt-Bereich (1 : 1000000000000) erhoben werden (ppt – parts per trillion = 1 Fremdatom auf 1 Billion Atome). Die Halbleitertechnik, insbesondere die Mikroelektronik, ist im Zuge dieser Entwicklung der Schrittmacher. Gleiche oder adäquate Forderungen werden zeitverzögert auch von anderen Zweigen, beispielsweise

der Pharmazie und der Lebensmittelindustrie, erhoben und durchgesetzt. Weltweit werden gegenwärtig etwa 35 Prozent der gesamten Industrieproduktion in Reinräumen durchgeführt, und von den insgesamt zirka 2 Millionen Quadratmetern Reinstraumfläche (clean room) sind etwa 50 Prozent in den Schaltkreisfabriken installiert.
Insgesamt sind im Halbleiterfertigungsprozeß 6 Kontaminationsquellen wirksam, die die ökonomische Effektivität des Fertigungsprozesses und die Qualität des Erzeugnisses (neben den ohnehin vorhandenen stochastischen und parametrischen Störquellen) beeinflussen. Diese Kontaminationsquellen sind:
– die in den Reinraum einströmende klimatisierte Luft
– das für den Prozeß erforderliche deionisierte Reinstwasser
– die für den Schichtaufbau und die Dotierung (Erzeugen der p- oder n-leitender Gebiete im Halbleiter) erforderlichen Dotier- und Schutzgase
– die im Prozeß erforderlichen Reinstchemikalien
– die im Prozeß eingesetzten Maschinen (technologische Spezialausrüstungen) und Ausrüstungen sowie Werkzeuge
– die im Prozeß tätigen (oder sich aufhaltenden) Menschen.
Diese Kontaminationsquellen haben unterschiedlichen Einfluß auf die Gesamtkontamination. Außerdem schwankt die Stärke des Einflusses zeitlich abhängig von den Entwicklungsergebnissen. Der Anteil einzelner Kontaminationsquellen ist in den Tafeln 1 und 2 zusammengestellt.

Tafel 1 Anteile der Kontaminationsquellen

	1986	1990	1995
Arbeitskräfte	[illegible]	[illegible]	[illegible]
Maschinen	22 %	28 %	32 %
Prozeßmaterial	25 %	30 %	20 %
Sonstiges	3 %	12 %	8 %

Tafel 2 Einflußfaktoren für die Kontaminationsquelle klimatisierte Luft

	1986	1990	1995
Luftführung	40 %	30 %	20 %
Luftfiltration	30 %	10 %	5 %
Cleanroomhülle	10 %	10 %	10 %
Potentialunterschiede	15 %	45 %	60 %
Material	5 %	5 %	5 %

Kontamination und Defektdichte

Der Maßstab für die hohen Anforderungen an die Umgebung ist die Defektdichte auf der Siliziumscheibe am Ende des Fertigungsprozesses (wafer process). Die Defektdichte gibt die Zahl von Partikeln auf einer bestimmten Fläche an. Die für hochintegrierte Schaltkreise erforderlichen Defektdichten liegen im Bereich von $< 0{,}1\,/cm^2$.
Die Gewährleistung einer niedrigen Defektdichte im Labor und in der Massenproduktion setzt extreme Anforderungen an die Reinräume voraus und zwingt gleichzeitig dazu, durch geeignete Maßnahmen dafür zu sorgen, daß die Kontamination
– möglichst nicht mit dem Werkstück, in diesem Fall der Siliziumscheibe, in Berührung kommt
– im Falle der Berührung keine haftende Verbindung eingeht.

Der Betreiber des Reinraums, das heißt der Entwickler und Produzent mikroelektronischer Bauelemente, muß seine Aufmerksamkeit im Zusammenhang mit der Defektdichte neben den Entwurfsbedingungen und -regeln (design rules) auf die Sauberkeit der Siliziumscheibe im Entwicklungs- und im Fertigungsprozeß konzentrieren. Neben physikalisch meßbaren Größen wirken jedoch auch hier subjektive Einflußgrößen, die nicht unterschätzt werden dürfen.
Gegenwärtig gibt es international für die Bestimmung der Kontamination und deren maximal zulässige Werte für alle 6 Kontaminationsquellen nur eine Vorschrift, die des US FED STD 209D, die im Grunde genommen von allen Ländern, die sich mit Reinraumtechnik befassen, original oder in Teilen verändert übernommen wurde. Dieser Standard definiert nach Klassen (class) entsprechend Bild 1 zwischen 1 und 100000. Daneben werden für Gase und Flüssigkeiten die Aussagen der ASTM herangezogen.
In der Summe bedeutet das jedoch, daß selbst bei Einhaltung der Reinraumklasse 1 (da nur die Luft definiert ist) ein Teilchen größer $0{,}3\,\mu m/ft^3$ zulässig ist. Da die Luft maximal 2 bis 5 Prozent Anteil an der Gesamtkontamination hat, sind noch immer – auch bei Klasse 1 – mehr als 1 Mio Partikel größer $0{,}3\,\mu m/ft^3$ wirksam (1 ft = 0,305 m).

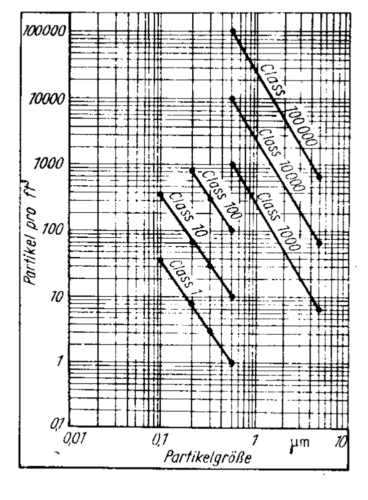


Bild 1 Reinraumklassen

Reduzierung der Defektdichte

Die Reduzierung der Defektdichte setzt die exakte Kenntnis der Quellen und der Einflußfaktoren voraus. Mittels erhöhten Luftstroms im Arbeitsbereich des Reinraums, Verringerung des elektrischen Wasserleitwertes bis an seine theoretische Grenze, extremer Reinigung der Gase und anderer Methoden wurde in der Vergangenheit versucht, die Kontamination an der Oberfläche der Siliziumscheibe zu reduzieren. Hierbei hat sich jedoch gerade in jüngster Zeit, im Zusammenhang mit Bezugspartikelgrößen von $< 0{,}5\,\mu m$, gezeigt, daß die Gravitation bei Partikelgrößen von $< 0{,}3\,\mu m$ keinen oder nur einen sehr geringen Einfluß auf die Partikelkontamination an der Scheibenoberfläche hat und daß die viel stärker wirkenden elektrostatischen Kräfte entscheidend davon beeinflußt werden, ob die Siliziumscheiben parallel oder senkrecht zum Luftstrom geführt werden. Untersuchungen in Japan /2/ haben ergeben, daß die Ablagerung von Partikeln, die $< 0{,}2\,\mu m$ sind, infolge elektrostatischer Aufladung relativ unabhängig von unterschiedlichen Luftgeschwindigkeiten (im Bereich von 10 bis 50 cm/s) erfolgt.
Hier wird deutlich, daß durch Reduzierung von Potentialunterschieden im Reinstraum bereits bei der Partikelablagerung eine entscheidende Defektreduzierung erreicht werden kann. Die beseitigten Potentialunterschiede verhindern gleichfalls willkürliche Entladungen und daraus resultierende Zerstörungen von Strukturen oder Schichten. Diese Beseitigung von Potentialunterschieden ist beispielsweise durch Deionisatoren im Luftstrom (sogenannte ESD – Electro-Static Devices), durch antistatischen Fußboden oder durch Antistatikkleidung erreichbar. An diesem Beispiel wird auch deutlich, daß neben den physikalisch meßbaren auch die subjektiven Größen wirksam werden können: Die Arbeitskraft kann die Vorkehrungen aufheben, indem sie zum Beispiel die antistatische Kleidung nicht trägt oder die antistatischen Handschuhe ablegt.
Weitere subjektive Faktoren wurden in /3/ ermittelt. Danach wurde für Arbeiten im Cleanroom im Bereich von Technologien für Strukturen von $< 1\,\mu m$ der sonst beim Umgang mit toxischen Gasen übliche Helm für alle Arbeitskräfte eingeführt. Trotz dieses Helms wurden in den Schichten noch immer folgende Werte gemessen:

Frühschicht	2 000 Partikel
Spätschicht	6 000 Partikel
Nachtschicht	1 000 Partikel

Die Partikelerhöhung in der 2. Schicht resultiert im speziellen Fall daraus, daß in dem dortigen Unternehmen in dieser Schicht die Wartung erfolgt. In der Nachtschicht, in der weniger Arbeitskräfte im Cleanroom sind, reduziert sich die Kontamination automatisch.

Arbeitsrichtungen und Entwicklungstendenzen

Kontaminationsreduzierung

Die notwendige Kontaminationsreduzierung zwingt zu konzentrierter Einflußnahme auf die einzelnen Kontaminationsquellen. Für die Kontaminationsquelle klimatisierte Luft wird an der weiteren Verbesserung der Luftfilter durch Einführung neuer Filterpapiere, neuer Faltbefestigungen, neuer Filterrahmendichtungen, verbesserter Cleanroom-Elemente usw. gearbeitet. Im Bereich der Reinstwassertechnik wird nach Einführung reverser Osmoseblöcke und emissionsarmer Kunststoffe, wie PVDF anstelle von PVC für Rohre und Armaturen, an der Ultrafiltration am *point of use* (Ort des Prozesses) gearbeitet. Für die Gasversorgungssysteme werden nach der Einführung verbesserter Filter und Edelstahlrohre in der Submikrometer-Technik elektropolierte Edelstahlrohre mit Oberflächenrauhigkeiten von $< 0{,}4\,\mu m$ eingeführt. Im Bereich der Chemikalienanwendung sind – neben den erhöhten Anforderungen an die Reinheit der Chemikalien – beim Anwender bereits ebensolche Anstrengungen bezüglich der Ab- oder Umfüllung und der Chemikalienversorgungssysteme wie beim Hersteller erforderlich. Die Zusammenstellung in der Tafel 1 zeigt jedoch, daß die größten Kontaminationsquellen die technologischen Spezialausrüstungen und die im Cleanroom sich aufhaltenden Menschen sind. Neben prinzipiellen konstruktiven Vorkehrungen an den Maschinen – Reduzierung der beweglichen Maschinen-

elemente, Vermeidung von Abrieb durch richtige Wahl der Elemente und Werkstoffe, keine bewegten Elemente über der Siliziumscheibe usw. – spielen hier der Systemaufbau, die hohe Verfügbarkeit (damit längere Betriebszyklen und verkürzte Wartungszyklen usw.) eine entscheidende Rolle. Auch in diesem Zusammenhang gilt, daß nur durch wissenschaftliche, gründliche und determinierte Untersuchungen und Analysen die richtigen Schwerpunkte definiert werden können, die es dann durch systematische Arbeit zu verbessern gilt.
Die Kontaminationsquelle *Mensch* läßt sich in ihrer Einheit von Qualifikation und Erfahrung, physiologischer und psychologischer Eignung sowie der Cleanroom-Kleidung durch systematisches Training mit dem geringsten Aufwand verbessern (siehe auch MP 5/1989, Seite 159).

Defektreduzierung

Die Arbeiten zur Defektreduzierung erfordern in erster Linie eine weitgehend perfekte Organisation und Leitung dieses Komplexes mit möglichst wenigen, aber sehr fähigen und uneigennützig arbeitenden Ingenieuren und Wissenschaftlern. Analysen bei Mitsubishi haben in den Prozessen mit Strukturgrößen von 1,5 bis 2,0 μm bezogen auf 100 % Defekte ergeben:
72,3 % Strukturdefekte durch eingebaute Partikel
15,2 % durch nicht eindeutig definierte Prozeßgrenzen
7,1 % durch nicht definierte Einflußfaktoren
3,0 % durch nicht ausreichend definierte Entwurfsgrenzen
2,4 % durch metallisierte Kontamination.
Nachgewiesenermaßen sind in neuen oder generell überholten Fertigungsstätten sowie nach Filtererneuerung oder Generalspülung der Leitungssysteme die Defekte sehr niedrig. Daraus leitet sich ab, daß eine gezielte Defektreduzierung eine vorherige Datenerfassung aller relevanten Prozeßorte erfordert. Nur unter dieser Voraussetzung und mit kürzesten Reaktionszeiten bereits bei sehr geringen Abweichungen ist eine gezielte Defektreduzierung möglich.

Literatur

/1/ Fukumoto, T.: Total contamination control management in semiconductor processing. 9th ICCCS[1], Los Angeles, Sept. 1988
/2/ Sakata, S.; Inone, M.; Chrifu, S.; Yoshida, T.; Okada, T.: Aerosol deposition on wafer surface. Sonderdruck
/3/ Martin, R. W.: Management of product defects in high technology programs contamination related defects. Lecture – IES[2]

[1] International Contamination Control Committee Symposium
[2] Institute of Environmental Sciences

Erfahrungen mit Modula-2-Compilern

Dr. Jürgen Lampe
Technische Universität Dresden, Sektion Mathematik

Die bereits in MP 10/88, S. 296 vorgestellte Programmiersprache Modula-2 stellt eine gelungene Verschmelzung von in Pascal bewährten Konzepten mit neueren Erkenntnissen der Programmier-Methodologie dar /1/. In diesem Zusammenhang ist insbesondere das – auch effektiv nutzbare – Modulkonzept hervorzuheben. Daneben bietet ein ebenso einfaches wie mächtiges Koroutinenkonzept, im wesentlichen erstmals seit SIMULA67, einem breiten Nutzerkreis die Möglichkeit, weitestgehend betriebssystemunabhängige Erfahrungen mit der Programmierung (quasi) paralleler Prozesse zu sammeln. Trotz der vorgenommenen Erweiterungen umfaßt der definierende Bericht in /2/ nur knapp 30 Seiten. Im Gegensatz zu Ada, dessen Syntax ebenfalls an Pascal angelehnt ist, weist Modula-2 dabei keine wesentlichen semantischen Differenzen auf, so daß die Anwendung für einen versierten Pascal-Programmierer allein aufgrund der Kenntnis des Berichts möglich ist.
Bei allem Positiven, was über Modula-2 zu sagen ist und was hierdurch nicht in Abrede gestellt werden soll, gibt es in der Praxis einige Probleme. Da das Ausmaß der daraus resultierenden Schwierigkeiten ganz wesentlich davon abhängt, inwieweit ihnen bereits in der konzeptionellen Phase Beachtung geschenkt wird, sollen hier einige meiner Erfahrungen dargestellt werden, die anderen Anwendern von Nutzen sein und möglicherweise aufwendige Überprüfungen ersparen können. Die Erfahrungen wurden im Rahmen der Entwicklung eines umfangreichen Programmsystems gewonnen, das eine offene Programmierumgebung zur Verwaltung und Ausführung dynamisch erzeugbarer Prozesse unter bestimmten Nebenbedingungen bildet /3/.

Compiler

Für Modula-2 ist inzwischen eine Reihe effektiver Compiler auf unterschiedlichen Rechnern verfügbar. Obgleich sich alle auf die Sprachbeschreibung von N. Wirth beziehen, sind es keine gleichen Sprachen, auch nicht wenn man den systemspezifischen Teil unberücksichtigt läßt. Dafür gibt es mehrere Ursachen, beispielsweise ist Modula-2 bislang nicht standardisiert.
Zwischen den definierenden Berichten in /2/ und /4/ bestehen Unterschiede (vgl. auch /5/). Je nachdem, zu welchem Zeitpunkt ein Compiler erstellt worden ist, wird deshalb auf einen anderen Bericht Bezug genommen. Obgleich die einzelnen Änderungen recht unerheblich erscheinen, kann der Aufwand erheblich sein, ihnen in existierenden Programmen Rechnung zu tragen. Einige dieser Differenzen werden noch genauer betrachtet werden.
Hinzu kommt, daß der Bericht einige Fragen offen läßt, die dann teilweise anhand des Verhaltens der an der Eidgenössischen Technischen Hochschule (ETH) Zürich gebauten Compiler beantwortet wurden.
Sowohl die internationale Standardisierungsorganisation (ISO) als auch das Britische Standardisierungsinstitut (BSI) haben Arbeitsgruppen gebildet, die gemeinsam an einem Standard für Modula-2 arbeiten. Die vorliegenden Veröffentlichungen /5/, /6/ lassen erwarten, daß auch Erweiterungen der Sprache erfolgen werden. Einige der diskutierten Modifizierungen finden sich bereits in Compilern, beispielsweise in /7/.
Die meisten Modula-2-Compiler erfordern einen eigenen Verbinder, weil nur dadurch die Konsistenz des Gesamtprogramms sicherzustellen ist. Dabei ist zu beachten, daß je nach System Speicher-Code, Objekt-Code oder ausführbare Programme entstehen. Damit wird über die Möglichkeiten zur Einbindung anderer Software und spezieller Leistungen des jeweiligen Betriebssystems (falls vorhanden, auch Overlays) weitgehend entschieden. Wenn es notwendig ist, umfangreiche Programmsysteme mit zahlreichen Modulen zu schreiben, ist die Existenz von Verwaltungshilfen (z. B. Make) nützlich. Sollte Protabilität zu den wesentlichen Kriterien zählen, so sind möglichst frühzeitig Tests relevanter Festlegungen mit unterschiedlichen Compilern vorzunehmen.

Syntaktische Differenzen

Da fast alle Differenzen auch einen syntaktischen Aspekt haben, werden hier nur diejenigen betrachtet, bei denen er überwiegt oder eine andere Zuordnung nicht sinnvoll ist.

```
IMPLEMENTATION MODULE myStorage;
(* Definitionsmodul entspricht Storage *)
  FROM SYSTEM IMPORT ADDRESS, WORD, ADR, SIZE;
  IMPORT Storage;

CONST Anz = 12; (* Anzahl der Verweisketten *)
      mS  = 4;  (* Min. Größe von Objekten
                   entspr. SIZE(ADDRESS)    *)
TYPE PADR = POINTER TO ADDRESS;
VAR stm : ARRAY [mS..Anz]OF PADR;
    i   : CARDINAL;

PROCEDURE ALLOCATE( VAR a: ADDRESS; s: CARDINAL);
 CONST EAnz = 8; (* Anzahl von gleichzeitig bereitzu-
                   stellenden Objekten  *)
 VAR i, j : CARDINAL;
     pa   : PADR;
     p    : POINTER TO ARRAY[0..100]OF WORD;
BEGIN
  IF s<=Anz THEN
    IF s<mS THEN s:= mS END;
    IF stm[s]=NIL THEN
      Storage.ALLOCATE(a, EAnz*s);
      p:= a;  pa:= a;
      (* Aufbau der Verweiskette: *)
      j:= 1;
      REPEAT
        pa^:= ADR(p^[j * s DIV SIZE(WORD)]);
        pa:= pa^;
        INC(j)
      UNTIL j= EAnz;
      pa^:= NIL
    ELSE
      a:= stm[s]
    END;
    (* Ein Objekt ausketten:*)
    pa:= a;
    stm[s]:= pa^.
  ELSE
    Storage.ALLOCATE(a, s)
  END
END ALLOCATE;

PROCEDURE DEALLOCATE(VAR a: ADDRESS; s: CARDINAL);
VAR i : CARDINAL;
    pa: PADR;
BEGIN
  IF s<=Anz THEN
    IF s<mS THEN s:= mS END;
    (* Freies Objekt einketten *)
    pa:= a;  pa^:= stm[s];    stm[s]:= a;    a:= NIL
  ELSE
    Storage.DEALLOCATE(a, s)
  END
END DEALLOCATE;

BEGIN (* Initialisierung: *)
  FOR i:= mS TO Anz DO stm[i]:= NIL END
END myStorage.
```

Bild 1

• **Exportliste im Definitionsmodul:** Ursprünglich gefordert, ist sie aber inzwischen durch die sinnvollere Festlegung, daß alle im Definitionsmodul spezifizierten Objekte automatisch exportiert werden, überflüssig geworden. Die Behandlung erfolgt unterschiedlich. /8/ gibt eine Warnung aus und /7/ eine Fehlermeldung, falls die Exportliste *vorhanden* ist, /9/ dagegen *fordert* sie.

• **Forward-Direktiven** sind eigentlich nicht Bestandteil der Sprache, wurden aber innerhalb des Projekts eines 1-Paß-Compilers wieder notwendig (vgl. auch /7/). Da nach /6/ die mit dem Kompilieren verbundenen Probleme ohnehin nicht in einem Durchlauf allgemein behandelbar sind, werden sie vermutlich nicht in den Standard aufgenommen werden.

• **Konstante Ausdrücke und strukturierte Konstanten:** Die Bildungsvorschriften für konstante Ausdrücke nach /4/ sind sehr restriktiv und werden teilweise durch (compilerbedingten) Ausschluß von Real-Ausdrücken noch weiter eingeschränkt. Es ist nicht möglich, solche compilerspezifischen Konstanten wie TSIZE(REAL) oder ORD('O') zu definieren. In /11/ ist eine veränderte Syntax gegeben, die diesen Mangel beseitigt. Unklar ist gegenwärtig noch, ob strukturierte Konstanten in den Standard aufgenommen werden und ob es einen Verkettungsoperator für Zeichenketten geben wird. (Strings, die sowohl' als auch" enthalten und soche, die Steuerzeichen enthalten, sind nicht als Konstanten definierbar.) /7/ erlaubt jedoch beides und zeigt damit die Möglichkeit dieser Erweiterung.

• Modula-2 läßt für Funktionen einen **beliebigen Ergebnistyp** zu. Es wird dadurch die Frage aktuell, Dereferenzierung und Selektion als spezielle Operationen zu behandeln. Für einen funktionalen Programmstil erscheint das sinnvoll. Das Problem wird diskutiert, wie das /6/ entnommene Beispiel zeigt.

```
TYPE arec = RECORD field1: INTEGER; field2:
INTEGER END; VAR a: ADDRESS; p: POINTER TO
arec; i: INTEGER;
(* assume that "a" has been set to a nice va-
lue *)
(* this is what we are forced to do at present *)
p: = a;
i: = p^.field1;
(* this is what I would like to do *)
i: = arec(a))^.field1;
```

Typumwandlungen

Modula-2 kennt zwei Arten von Typumwandlungen: Konvertierungen und Casts (Uminterpretationen). Dabei gelten Konvertierungen, die durch spezielle Prozeduren erfolgen, als sogenannte sichere Umwandlungen, beispielsweise CHR(70), VAL (CHAR,70), FLOAT (4), TRUNC(7.4). Zu beachten ist dabei, daß das Ergebnis von TRUNC entweder INTEGER, CARDINAL oder umgebungsabhängig sein kann.

Casts werden zwar allgemein als unsichere und überdies compilerabhängige Typumwandlungen betrachtet, sind in Verbindung mit Pointern und verdeckten Typen aber durchaus sicher und nützlich einsetzbar. Störend macht sich dabei bemerkbar, daß Casts ihrer Form nach Ausdrücke sind und demzufolge nicht für VAR-Parameter stehen und nicht dereferenziert oder selektiert werden können. Das erfordert jeweils explizite Zuweisungen zu eigentlich überflüssigen Hilfsvariablen.

Zirkularitäten

Die zusätzlichen Ausdrucksmittel von Modula-2 haben auch neue Möglichkeiten rekursiver Abhängigkeiten geschaffen. Während wechselseitiger Import in Definitionsmodulen bei inneren Modulen zumindest prinzipiell noch behandelbar ist (wenn auch nicht von allen Compilern), ergeben sich für Kompilationseinheiten unlösbare Probleme, da keine korrekte Kompilationsreihenfolge mehr angegeben werden kann. Zyklen im Abhängigkeitsgraphen der Definitionsmodule müssen deshalb vermieden werden; möglicherweise auch durch einen zusätzlichen Modul, der die kritischen Objekte definiert.

Eine weitere unzulässige Zirkularität zeigt das folgende Beispiel:

```
TYPE not Possible = RECORD proc: PROCEDURE
                    (VAR not Possible); ...
                    END
```

In diesem Fall hilft nur das Ausweichen auf einen Zeigertyp.

Der Modul SYSTEM

In Modula-2 werden sogenannte systemabhängige Möglichkeiten durch den Pseudomodul *SYSTEM* bereitgestellt. Module, die Objekte von SYSTEM importieren, werden als Hardwaremodule betrachtet. Bei genauerer Überprüfung zeigt sich jedoch, daß diese Charakterisierung nicht den tatsächlichen Verhältnissen entspricht. Der Modul SYSTEM exportiert sehr unterschiedliche Objekte, von denen einige durchaus systemunabhängig sind oder zumindest so verwendet werden können. Für die Entwicklung offener Programmiersysteme ist die Verwendung von ADDRESS und WORD sogar unbedingt erforderlich, weil darin die einzige Möglichkeit besteht, in gewisser Weise typunabhängige (generische) Prozeduren zu formulieren. Beispielsweise ist die Funktion

```
PROCEDURE IsNIL(a: ADDRESS): BOOLEAN;
BEGIN RETURN a = NIL END IsNIL
```

weder hardwarenah noch systemabhängig. Ähnliches gilt für andere von SYSTEM exportierte Objekte. Die völlig unnötige Unbestimmtheit des Berichts in diesem Punkt wirkt ausgesprochen hinderlich beim Erstellen portabler Software. Statt der Festlegung, daß ADDRESS und CARDINAL kompatibel sind, was auf dem 8088 praktisch nicht zutreffend ist, wäre eine genau definierte Adreßarithmetik erforderlich, wobei INC, DEC und Differenz allgemein als ausreichend angesehen werden. Es fehlt eine Definition der Beziehung zwischen ADDRESS und SIZE bzw. TSIZE. Fast alle vorliegenden Compiler messen den Speicherbedarf in Adreßschritten, so daß die im Bericht als Beispiel enthaltene Speicherverwaltung mit Allocate(p, SIZE(p^)) aufgerufen werden könnte. Andererseits legt der Bericht nahe, TSIZE(WORD)=1 zu wählen, was in /8/ enthalten ist.

Weitere Probleme ergeben sich mit dem universellen Wertparametertyp ARRAY OF WORD. Compiler mit Beschränkungen hinsichtlich der Ergebniswerte von Funktionen erlauben für derart offene Parameter im allgemeinen keine echten Ausdrücke, wie Funktionsaufrufe. Manchmal ist der Typ REAL ausgeschlossen, weil die Übergabe in einem speziellen Akkumulator erfolgt.

Ein Weg, diesen Problemen zumindest teilweise auszuweichen, besteht darin, einen eigenen, systemspezifischen (besser: compilerspezifischen) Modul zu programmieren und, soweit möglich, von diesem statt von SYSTEM zu importieren. Portierungen erfordern dann lediglich eine Anpassung dieses Moduls (siehe folgendes Beispiel).

```
DEFINITIONS MODULE MySystem;
IMPORT SYSTEM;
(* EXPORT QUALIFIED PROCESS, SizeWord,
                    SizeAddr, SizeFac,
                    DECAddr, INCAddr,
                    DIFAddr; *)
CONST SizeWord = 2; (* = SYSTEM.SIZE
      (WORD) *)
      SizeAddr = 4; (* = SYSTEM.SIZE
      (ADDRESS) *)
      SizeFac = 1; (* Relation between SIZE
      and ADDRESS *)
TYPE PROCESS = SYSTEM.PROCESS; (*or SY-
STEM.ADDRESS *)
PROCEDURE DECAddr (VAR A: SYSTEM.AD-
DRESS; B: CARDINAL);
PROCEDURE INCAddr (VAR A: SYSTEM.AD-
DRESS; B: CARDINAL);
PROCEDURE DIFAddr (A, B: SYSTEM.ADDRESS;
VAR H,L: CARDINAL);
END MySystem.
```

Bei Verwendung von Koroutinen ist es sinnvoll, eine Konstante für den damit verbundenen Overhead zu definieren, da die Unterschiede zwischen einzelnen Compilern erheblich sind. Es ist auch zu berücksichtigen, daß die Bereichsangabe in NEWPROCESS teilweise in Word, meist jedoch in Byte erfolgt. Die Festlegungen zur Quasiparallelität sind im Bericht sehr vage gehalten. Daraus und aus den engen Beziehungen zum jeweiligen Betriebssystem folgt, daß es in diesem Punkt die größte Variationsbreite zwischen den Compilern gibt. Wesentliche Informationen, wie über den tatsächlich benötigten Arbeitsspeicher oder dessen Verschieblichkeit (z. B. für ein Software-Paging), sind im allgemeinen nicht verfügbar. Um trotzdem eine hohe Portabilität zu erreichen, ist eine disziplinierte (eingeschränkte) Verwendung erforderlich. Bei gegebenem Zielsystem sollte (falls möglich) die Auswahl eines Compilers sehr sorgfältig und unter Zuhilfenahme eigener Testbeispiele erfolgen, da diese Eigenschaften meist nur unzureichend dokumentiert sind.

Der Modul Storage

Ein Modul *Storage*, der zumindest die Prozeduren ALLOCATE und DEALLOCATE exportiert, wird als zu jedem Compiler gehörig vorausgesetzt. Prinzipiell kann zwar jeder Nutzer auch einen eigenen Modul zur Speicherverwaltung ergänzen, aber zuweilen ist das nicht mehr auf der Modula-2-Ebene möglich, da spezielle Betriebssystemfunktionen dazu erforderlich sind.

Im Gegensatz zur M2RT11 /12/ enthalten fast alle bekannten Compiler für 16-Bit-PCs Einschränkungen bezüglich der zulässigen Parameter beim Aufruf von DEALLOCATE, und zwar darf in der Regel nur ein zuvor mit ALLOCATE zugewiesener Bereich freigegeben werden. Dabei wird entweder die gleiche Größenangabe gefordert oder der Wert des zweiten Parameters ignoriert; eine teilweise Freigabe ist also nicht möglich.

Die Speichervergabe erfolgt nach unterschiedlichen Strategien entweder mit oder ohne Wiederzusammenfassung freier Bereiche. Gemeinsam ist aber allen diesen Systemen, daß sie mit dem Overhead zur Verwaltung verbunden sind, beispielsweise 4 oder 5

Fortsetzung auf S. 307

Turbo-Pascal-Praxis

Teil 3

Manfred Zander, Dresden

Wir haben uns im Teil 2 unseres Kurses in MP 8/89 der Programmierung der Fenstertechnik für den PC 1715 und dazu kompatibler Rechner gewidmet. Auf den folgenden 4 Seiten sollen neben der Arbeit mit Menü- und Hilfsausschriften sowohl Hardcopy-Funktionen als auch Prozeduren zur Anzeige des Disketten-Directories dargeboten werden.

Bildschirm-Darstellungen

In vielen Programmen, die für Nutzer gedacht sind, die noch nie etwas mit Rechentechnik zu tun hatten, ist der Programmierer gezwungen, für jede noch so kleine Eingabeoperation einen ganzen Bildschirm voller Kommentartexte zu schreiben, der eine möglichst optimale Bedienerführung gewährleistet. In solchen Programmen kommen oft drei, vier oder noch mehr write oder writeln-Befehle auf jeden anderen. Bei der Austestung dieser Programme sind die Kompilierzeiten dann wieder so groß, daß der Turbo-Effekt nicht mehr zur Geltung kommen kann. Durch die vielen Ausgabebefehle wird der Quelltext so gewaltig, daß er in eine Vielzahl von Include-Files aufgegliedert werden muß. Das Editieren wird schon bei kleinsten Änderungen sehr mühsam.

Liegt ein solcher Fall vor, kann die Lösung nur darin liegen, die Bildschirm-Darstellungen für die Zeit der Programmerarbeitung völlig vom übrigen Programm zu trennen. Ist das Programm fertig, kann man sie dann wieder in den Bestand des Programmtextes aufnehmen. In diesem Abschnitt wird nun eine solche Vorgehensweise dargestellt. Besonders günstig ist die maschinelle Einarbeitung der Bildschirmdarstellungen in den Programmtext. Ein Nachteil ist der etwas langsamere Bildaufbau in der Testphase des Programms. Dieser Nachteil kann aber sicher toleriert werden.

Das Prinzip der Lösung besteht in der Auslagerung der Bildschirmdarstellungen in eine Diskettendatei. Jeder Bildschirm hat in dieser Datei eine Kennung, unter der er aufgefunden werden kann. Der Aufruf der Bildschirmdarstellungen erfolgt über eine Prozedur, der diese Kennung übergeben wird. In Bild 23 wird ein Auszug aus einer solchen Bildschirmdatei angegeben. Die Prozedur BildWrite (Bild 24), welche die Bildschirmdarstellungen aus der Datei liest und auf den Bildschirm bringt, realisiert folgende Einzelaktionen. Als erstes wird der Bildschirm gelöscht und die Bildschirmdatei auf ihren Anfang zurückgesetzt. Anschließend werden solange Zeilen aus dieser Datei gelesen, bis die zu suchende Kennung des zu schreibenden Bildes gefunden wird. Die folgenden Zeilen in der Datei enthalten nun das gesuchte Bildschirmaussehen. Diese Zeilen werden nun einzeln aus der Datei gelesen und auf den Bildschirm geschrieben. Als Endekennzeichen eines Bildes dient dabei die Kennung des nächsten Bildes oder das physische Dateiende. Wurde ein Bild gesucht, welches nicht in der Datei enthalten ist, erhält der Bediener eine entsprechende Mitteilung.

Durch die Hilfe dieser Prozedur genügt es jetzt, im Programm nur eine Anweisung zu schreiben, um dem Bildschirm das in der Datei gespeicherte Aussehen zu geben. Da diese Lösung nur dann angewendet wird, wenn in der Datei eine Vielzahl von Bildern gespeichert werden muß, ist die Datei natürlich auch sehr groß. Das heißt, die Suchzeiten für ein Bild sind teilweise recht lang. Der Vorteil liegt in der Tatsache, daß im Programm die Anweisung

bildwrite('Plausibilität');

genügt, um dem Bildschirm das gewünschte Aussehen zu geben. Weiterhin ist es nun auch möglich, ohne Umschreiben und Neu-Kompilieren des Programmes, die Textdarstellungen zu ändern. Es ist sogar denkbar, die kommentierenden Texte von den ersten Nutzern des Programms, die ja die Spezialisten auf ihrem Gebiet sind, ändern zu lassen, was mit jedem Textverarbeitungssystem möglich ist, da die Bilddatei ein reines Textfile ist. Sind die Tests des Programmes abgeschlossen, sollte diese Bildschirmdatei natürlich wieder in den Programmtext aufgenommen werden, um eine kompakte Einheit der Lösung zu erreichen. Dateien, an deren Namen sich später niemand mehr erinnert, werden leicht gelöscht, und unser Programm wird zum Invaliden. Dazu sind zwei Maßnahmen notwendig. Zum einen muß unsere Bilddatei in eine Pascal-Quelldatei umgewandelt werden, die in das Programm hineincompiliert werden kann. Zum zweiten muß die Prozedur bildwrite geändert werden. Diese mühselige Arbeit kann uns aber der Rechner abnehmen. Wir schreiben uns dazu ein Programm, das diesen Quelltext schreibt. Das Programm Transformation (Bild 25) liest unsere alte Bilddatei und erstellt uns sowohl die neue Prozedur bildwrite (Bild 26) als auch die Prozeduren, die später die einzelnen Bildschirmdarstellungen erstellen sollen (Bild 27). Die Namen der neuen Prozeduren werden in diesem Programm aus den alten Kennungen gebildet. Um Konflikte mit eventuell bereits bestehenden Bezeichnern zu vermeiden, wird den Prozedurnamen als erstes Zeichen noch ein **x** vorangestellt. Die Funktion des Programms Transformation ersieht man am einfachsten aus den gebildeten neuen Dateien.

Die Datei mit dem Namen BILDWRIT.INC enthält nach diesem Programm die neue Prozedur bildwrite (Bild 26). In der zweiten von unserem Programm erstellten Datei mit dem Namen NDEF000.INC sind nun alle die Prozeduren enthalten, die von der neuen Prozedur bildwrite aufgerufen werden (Bild 27). Die Tatsache, daß diese Prozeduren alle als Overlay-Prozeduren gekennzeichnet wurden, liegt an der Spezifik des hier genutzten Programmbeispiels.

```
In dieser Datei sind alle Bildschirm Masken des Programmes NDEF
enthalten.
Jede Maske beginnt mit ihrer Kennung :   >>>aaaaaaaa
Diese Kennung muss linksbuendig am Anfang jeden Bildes stehen!
>>>Hauptmenue

 ------------------------------------
!          H a u p t m e n u e       !    (= N D E F =)
!====================================!  ______=======___________________
!BE-Typen      neu  definieren   !  1 !!          S t e u e r u n g        !
!              alte aendern      !  2 !!-----------------------------------!
!BE auf LP     neu  setzen       !  3 !!  Umfang der Plausibilitaets-   T  
!              verschieben       !  4 !!  tests voreinstellen          Tr!
   ....
   ....
>>>Plausibilitaet
Folgende Stufen der Plausibilitaetstests sind moeglich:
    ----------------------------------------------------------  ------
Es erfolgt keinerlei Pruefung (schnellste Mode )     :        0

Pruefung bei Setzen ob BE-Bezugspunkt auf LP         :        1
    ...
    ...
>>>Freiflaechen
                                ---------------------------------------------
                               !     Links/unten      !     Rechts/oben     !
Hier koennen Aussparungen      !----------------------!---------------------!
auf der Leiterplatte defi-     !____X____!____Y____  !____X____!____Y_____!
   ....
```

◀ **Bild 23** *Datei mit Bildschirmausschriften*

```
procedure BildWrite(suchtxt:string80);
  var zeile:string80;
  begin
    clrscr;                                    (* Bildschirm loeschen  *)
    reset(BildDatei);                          (* Bilddatei auf Anfang *)
    repeat
      readln(BildDatei,zeile)                  (* Suchen der Kennung   *)
    until (zeile='>>>'+suchtext) or (eof(BildDatei));
    if not(eof(BildDatei)) then
      begin
        readln(BildDatei,zeile);
        repeat                                 (* Beschreiben des Bild-  *)
          writeln(zeile);                      (* schirmes aus der Datei *)
          readln(BildDatei,zeile)
        until (copy(zeile,1,3)='>>>') or (eof(BildDatei))
      end
    else write('Nicht gefunden');
  end;
```

Bild 24 Prozedur zum Schreiben der Bildschirmausschriften

```
program Transformation;
const Hochkomma = #39 ;
var BildDatei,NeuInc,NeuBiW:text;
    zeile:string[80];
begin
  assign(BildDatei,'ndeftx*.txt');reset(BildDatei);
  assign(NeuInc,'ndef000.inc');rewrite(NeuInc);
  assign(NeuBiW,'b:'Anrst.inc');rewrite(NeuBiW);
  writeln(NeuBiW,'procedure bildwrite(suchtxt:string80);   (* Neuer     *)
  writeln(NeuBiW,'  begin');                                 (* Programm  *)
  writeln(NeuBiW,'    clrscr;');                             (* Kopf      *)
  repeat
    readln(BildDatei,zeile)        (* Suchen nach erstem Bildschirm-Bild *)
  until copy(zeile,1,3)='>>';
  repeat
    zeile:=copy(zeile,4,length(zeile)-3);
    writeln(NeuBiW,
          if suchtxt='+Hochkomma+zeile+Hochkomma+' then x'+zeile+';');
    writeln(NeuInc,'overlay procedure x'+zeile+';');
    writeln(NeuInc,' begin');
    writeln('procedure x'+zeile);   (* Meldung ueber Programmlauf *)
    readln(BildDatei,zeile);
    repeat
      writeln(NeuInc,'  writeln('+Hochkomma+zeile+Hochkomma+');');
      readln(BildDatei,zeile)
    until (copy(zeile,1,3)='>>') or eof(BildDatei);
    writeln(NeuInc,'end;')
  until eof(BildDatei);
  writeln(NeuBiW,'  end;
  close(BildDatei);
  close(NeuInc);
  close(NeuBiW);
end.
```

Bild 25 Quelltext-schreibendes Programm

```
overlay procedure xHauptmenue;
 begin
  writeln(' .................................. ');
  writeln(' |          H a u p t m e n u e     |   * N D E F *');
  writeln(' |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^|  ________ ');
  writeln(' |BE-Typen      neu  definieren   | | |      S t a r t r i n g ');
    ...
    ....
 end;
overlay procedure xPlausibilitaet;
 begin
  writeln('');
  writeln('Folgende Stufen der Plausibilitaetstests sind moeglich:');
  writeln('^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^');
    ....
    ...
 end;
 ...
 ...
```

Bild 27 Automatisch erstellter Quelltext

Hat der Programmierer das Programm *Transformation* auf seine Textdatei mit den Bildschirmdarstellungen angewendet, sind nur noch wenige Schritte notwendig. Als erstes tauscht er die alte Prozedur BildWrite gegen die neue aus, was mit dem Editor-Kommando ^KR (Block-Lesen) kein Problem ist. Dann fügt er im Programm die Compiler-Anweisung **(* $I ndef000.inc *)** ein. Nach der nun notwendigen Neuübersetzung des Programmes ist alles erledigt.

Als letztes sollte noch gesagt werden, daß dieses Programm nur *ein* Beispiel ist. In vielen Situationen ist es möglich, stupide Programmierarbeit von kleinen, selbstgeschriebenen Hilfsprogrammen erledigen zu lassen. Wem dieser Gedanke vertraut geworden ist, dem ergeben sich in der täglichen Arbeit oft solche Möglichkeiten.

Daten – und Dateiarbeit

In diesem Kapitel soll es sowohl um die Bearbeitung von Daten aus Turbo-Pascal-fremden Diskettenfiles als auch um die Verwaltung von Daten in listenartigen Strukturen im Hauptspeicher gehen. Neben diesen größeren Problemen soll hier aber auch Platz für einige kleinere sein. So wird eine Möglichkeit geboten, das Verzeichnis einer Diskette selektiv zu lesen.

Hardcopy

Hat ein Rechner bereits eine Hardcopy-Funktion, so wird diese oft und gern genutzt. Der Griff zur Taste PrnScr (PrintScreen) wird zu einem wertvollen Hilfsmittel in der Arbeit. Vom Bildschirm schnell und problemlos ein interessantes Aussehen auf dem Papier festhalten zu können, effektiviert die Arbeit an einem Rechner doch sehr. Hat man sich an einem Rechner höherer Leistungsklasse an dieses Hilfsmittel gewöhnt, vermißt man es auf allen anderen. Hier möchte ich nun zwei Prozeduren anbieten, die diesen Service auch auf kleineren Rechnern ermöglichen und speziell für den PC 1715 entwickelt wurden. Für ihre Lauffähigkeit sind ähnliche Einschränkungen wie für die Fensterhilfen gültig. Der Bildwiederholspeicher muß linear von der Hauptspeicheradresse F800H an aufgebaut sein, und jede Position auf dem Bildschirm muß genau einer Speicherzelle entsprechen. Dies ist zum Beispiel nicht mehr der Fall, wenn Bildschirmsteuerzeichen wie im Kapitel Auswahlmenü Speicherplatz beanspruchen. Eine Sicherung von Bildschirmen mit intensiven oder inversen Bereichen ist also nicht möglich.

```
procedure bildwrite(suchtxt:zeilentyp);
 begin
   clrscr;
    if suchtxt='Hauptmenue' then xHauptmenue;
    if suchtxt='Plausibilitaet' then xPlausibilitaet;
    if suchtxt='BETypendefinieren' then xBETypendefinieren;
       ....
    if suchtxt='Freiflaechen' then xFreiflaechen;
       ....
    if suchtxt='Anzeigen' then xAnzeigen;
 end;
```

Bild 26 Neue Prozedur BildWrite

```
procedure List_Hardcopy;
  var Bild : array[1..24,1..80] of char absolute $F800;
      i,j  : byte;
  begin
    for i:=1 to 24 do
      begin
        for j:=1 to 80 do write(lst,Bild[i,j]);
        writeln(lst);
      end;
  end;
```

Bild 28 Die einfache Hardcopy-Prozedur

Als erstes wird die Prozedur List_Hardcopy vorgestellt, die das aktuelle Aussehen des Bildschirmes ausdruckt (Bild 28).

Die Wirkungsweise dieser Prozedur beruht in der Überlagerung des Arrays *Bild* mit dem Bildwiederholspeicher des Rechners. Die Ausgabe des Bildschirmes auf den Drucker erfolgt zeilenweise, Zeichen für Zeichen. Diese Prozedur beansprucht in unseren Programmen kaum Platz. Konkret sind es 10 Byte Daten und 125 Byte Code.

Wir wollen aber noch eine zweite, komfortablere Hardcopy-Prozedure vorstellen, mit der es auch möglich ist, den Bildschirminhalt in eine Diskettendatei zu schreiben, wodurch er gesichert wird und dann jederzeit wieder zu Papier gemacht werden kann. Diese Programmlösung wird auch deshalb angeführt, da in ihr gut zu erkennen ist, wie man das Standardtextfile LST als File-Variable behandeln kann. Dieses Prinzip ist auch in vielen anderen Situationen interessant. Selbstverständlich benötigt diese Variante mehr Platz. Sie belegt genau 283 Byte Daten und 422 Byte Code. (Bild 29)

Die prinzipielle Wirkungsweise der Prozedur List_Hardcopy finden wir hier in der lokalen Prozedur AusgabeAuf wieder. Worauf diese Ausgabe erfolgen soll, wird der Prozedur als Parameter übergeben. Dieser Parameter spezifiziert ein beliebiges Text-File. Auch der Drucker, also das Standard-File LST, ist ein Text-File.

Die Prozedur hardcopy sichert zuerst das Aussehen der ersten Bildschirmzeile in einem temporären Feld. Dann erfolgt auf dieser ersten Bildschirmzeile die Frage nach dem Namen des Diskettenfiles, unter dem die Hardcopy abgelegt werden soll. Anschließend wird das Aussehen der ersten Zeile wieder hergestellt und der Kursor ausgeschaltet. Ist der eingegebene Name leer, daß heißt, die Frage wurde nur mit der Entertaste beantwortet, so erfolgt die Ausgabe wie gewohnt über den Drucker. Ansonsten wird die angewiesene Datei zum Schreiben vorbereitet und gefüllt. Zum Abschluß muß natürlich der Kursor wieder eingeschalten werden.

```
procedure hardcopy;
  const Kursoron = #$82;
        Kursoroff= #$83;
  var Bild:array[1..24,1..80] of char absolute $F800;
      i,j :byte;
      Temp:array[1..80] of char;
      Dateiname:string[14];
      Diskfile :text;
  procedure AusgabeAuf(var Geraet:text);
    begin
      for i:=1 to 24 do
        begin
          for j:=1 to 80 do write(Geraet,Bild[i,j]);
          writeln(Geraet);
        end;
      close(Geraet)        (* close auf lst bleibt ohne Wirkung *)
    end;
  begin
    for i:=1 to 80 do           (* Sichern der ersten Zeile des BS  *)
      Temp[i]:=Bild[1,i];       (* in ein temporaeres Feld          *)
    gotoxy(1,1);
    write('Auf Datei (RET fuer Drucker):');
    read(Dateiname);            (* lesen File-Namens unter dem die  *)
    write(kursoroff);           (* Hardcopy gespeichert werden soll *)
    for i:=1 to 80 do
      Bild[1,i]:=Temp[i];       (* Wiederherstellen erste BS Zeile  *)
    if dateiname<>'' then       (* Test des Dateinamens *)
      begin
        assign(Diskfile,dateiname);
        rewrite(Diskfile);
        AusgabeAuf(Diskfile)   (* Ausgabe auf Diskfile  *)
      end
    else AusgabeAuf(lst);       (* Ausgabe auf Drucker   *)
    write(kursoron);
  end;
```

◀ *Bild 29 Hardcopy wahlweise auf Drucker oder Datei*

```
type string11=string[11];
procedure MiniDir(LW:Char;Maske:string11);
  var fcb  : array[0..35] of char absolute $5C;
      dma  : array[0..127] of byte;
      a,i  : byte;
  procedure Eine_Eintragung(DMA_Position:integer);
    var Name:string[12];
    begin
      DMA_Pos:=DMA_Pos*32+1; Name:='';
      for i:=0 to 11 do
        Name:=Name+chr(dma[DMA_Pos+i] and $7F);
      insert('.',Name,9);
      write(Name,' | ');
    end;
  begin
    LW:=upcase(LW);
    bdos($1A,addr(dma));                        (* Einstellen DMA Adresse   *)
    fcb[0]:=chr(integer(LW)-integer('A')+1);    (* Einstellen des Laufwerks *)
    fcb[12]:=#0 ;
    a:=bdos($11,addr(fcb)) ;                    (* Suchen nach erster Eintragung *)
    while a <> 255 do
      begin
        Eine_Eintragung(a) ;
        a:=bdos($12,addr(fcb)) ;                (* Suchen weiterer Eintragungen *)
      end;
    writeln;
  end;
```

Bild 31 Kleines Disketten-DIRectory MiniDir

Existiert bereits eine Disketten-Datei unter dem angegebenen Namen, so ist deren ursprünglicher Inhalt gelöscht. Da dieses Problem keinesfalls spezifisch für diese Prozedur ist, wollen wir eine kleine Funktion angeben, die abprüft, ob eine Datei bereits existiert. Jeder Programmierer kann sie nutzen, wenn Dateinamen überprüft werden müssen (Bild 30).

Wollen wir die Dienste der Funktion Exist in Anspruch nehmen, sieht der Test des Dateinamens in obiger Prozedur folgendermaßen aus:

```
...
if (dateiname<>'')and
   (not(exist(dateiname)) then
...
```

Diskettenverzeichnisse

In Fortführung des letzen Abschnittes wollen wir noch ein wenig bei den Namen von Dateien bleiben. Wird der Nutzer eines Programmes aufgefordert, einen Dateinamen anzugeben, ist dieser meistens etwas allein gelassen. Kann er sich noch an den Namen *seiner* Datei erinnern, so nicht mehr an die Extension. Weiß er diese noch, fehlt ihm der Name. Oder aber er weiß im Augenblick keines von beiden. Der Programmierer sollte an diesen Programmstellen Unterstützung vorsehen. Bei Compilern, die den Systembefehl DIR unterstützen, ist dies leicht möglich. Hier nun sollen zwei Prozeduren auch für die kleineren Compiler vorgestellt werden. Sie sind auf allen CP/M-kompatiblen Betriebssystemen lauffähig, da sie die Diskettenzugriffe über BDOS-Funktionen realisieren. Als erstes stellen wir die Prozedur MiniDir vor. Eine genaue Erläuterung der Wirkungsweise der verwendeten BDOS-Rufe muß hier entfallen, da dies den Rahmen des Kurses sprengen würde. Es sei nur erwähnt, daß das Array **fcb** einen File-Control-Block darstellt. Dieses Array wird vor allem dazu genutzt, dem Betriebssystem Informationen zu übergeben. Im Array **dma** finden wir nach den BDOS-Rufen die angeforderten File-Namen. Die BDOS-Funktionen selbst übergeben lediglich eine Information, wo im Array dma der neue File-Name zu finden ist, da im dma-Block bis zu vier Namen gleichzeitig geführt werden können. Haben diese Funktionen keine weitere Verzeichniseintragung gefunden, übergeben sie den Wert 255 an das Programm. Der Anweisungsteil der Prozedur MiniDir stellt die Vorgabewerte des Arrays fcb ein und sucht nach den Verzeichniseintragungen. Die lokale Prozedur Eine_Eintragung, der die Position des neuen File-Namens übergeben wird, schreibt diesen Namen auf den Bildschirm (Bild 31).

Der Aufruf der Prozedur MiniDir erfordert eine gewisse Sorgfalt. Es handelt sich wirklich nur um ein *Mini*-Directory. Werden nicht vorhandene Laufwerke im Prozedure-Kopf angegeben, kommt es zu Systemabstürzen. Auch die Maske muß mit genau 11 Zeichen aufgefüllt werden. Sie stellt eine Gültigkeitsmaske für das Suchen nach Verzeichniseintragungen dar. Die ersten 8 Zeichen selektieren die Dateinamen, das 9., 10. und 11. Zeichen selektieren die Extension. Ein Punkt darf in der Maske nicht vorkommen. Für beliebige Zeichen ist je ein "?" einzutragen. Soll die Länge der Namen begrenzt werden, so

```
function exist (filename:string[14]):boolean;
  var tfile:file;
  begin
    assign(tfile,filename);
    {$I-}
    reset(tfile);
    {$I+}
    if ioresult <> 0 then
      exist:=false
    else exist:=true;
    close(tfile);
  end;
```

Bild 30 Test auf Existenz einer Datei

sind die restlichen Zeichen mit Leerzeichen aufzufüllen. In Bild 32 sind einige Beispiele für den Aufruf der Prozedur MiniDir angegeben.

Die Prozedur MiniDir listet die gefundenen Verzeichniseintragungen von der aktuellen Kursorposition aus auf. Es werden alle in die Maske passenden Dateien aufgelistet, also auch Dateien mit den Attributen SYS und R/O. Stand der Kursor an einem Zeilenanfang, stehen immer 5 Verzeichniseintragungen nebeneinander.

Als nächstes wird eine Lösung vorgestellt, die viele der aufgeführten Unzulänglichkeiten nicht mehr aufweist. Die Angabe des Parameters VerzMaske der Prozedur DIR kann nun dem Nutzer überlassen werden (Bild 33).

Die Prozedur DIR selbst bearbeitet und interpretiert die angegebene Verzeichnismaske und selektiert, falls angegeben, die Laufwerksangabe. Fehlt diese, wird das aktuelle Laufwerk ermittelt. Die Wirkungsweise der Funktion HoleDirectory entspricht der der Prozedur MiniDir. Sie übergibt die Anzahl der aufgefundenen Verzeichniseintragungen. Die Angabe der Verzeichnismaske kann nun recht frei erfolgen. Zwischen kompletten Dateinamen mit Laufwerksangabe und einer völlig leeren Zeichenkette ist alles erlaubt. Die Interpretation der Verzeichnismaske in dieser Prozedur weicht gewollt von der Interpretation der Maske beim Systembefehl DIR ab. So liefert zum Beispiel der Aufruf:

DIR(a:TP.A);

alle die Dateien, deren Name mit TP. und deren Extension mit A beginnt. Dies war in der konkreten Anwendung, der diese Lösung entnommen wurde, gefordert. Ist in einer anderen Anwendung eine andere Interpretation gewünscht, kann dies leicht geändert werden. Im Gegensatz zum MiniDir werden die Verzeichniseintragungen in dieser Lösung nicht nur auf den Bildschirm übertragen, son-

```
MiniDir('A','????????ASC');  (* alle Dateien mit Ext. ASC auf LW A      *)
MiniDir('B','TURBO??????');  (* Dateien die mit TURBO beginnen Ext. bel.*)
MiniDir('A','TURBO   ???');  (* Dateien Namens TURBO mit bel. Extension *)
MiniDir('A','???     ASC');  (* Dateien mit hoechstens dreistelligem    *)
                             (* Namen und Extension ASC auf LW A        *)
MiniDir('B','???????????');  (* alle Dateien auf LW B                   *)
```

Bild 32 Aufruf-Beispiele für MiniDir

```
type string11=string[11];string14=string[14];
var DateiNamen : array[1..128] of string[12];
    Datei_Anzahl: byte;
function HoleDirectory(LW:Char;Maske:string11):byte;
  var fcb  : array[0..35] of char absolute Maske;
      dma  : array[0..127] of byte;
      a,i,zaehler  : byte;
  procedure Eine_Eintragung(DMA_Posi:integer);
    begin
      DMA_Posi:=DMA_Posi*32+1 ;
      if (dma[DMA_Posi+10] div 128 =0)              (* Test auf gesetzte    *)
          and(dma[DMA_Posi+11] div 128 =0)  then (* Atribute Sys und R/O *)
        begin
          DateiNamen[zaehler]:='';
          for i:=0 to 11 do
            DateiNamen[Zaehler]:=
            DateiNamen[Zaehler]+chr(dma[DMA_Posi+i] mod 128);
          insert('.',DateiNamen[Zaehler],9);
          zaehler:=succ(zaehler)
    end end;
  begin
    zaehler:=1 ;LW:=upcase(LW);
    bdos($1A,addr(dma)) ;                (* Einstellen DMA-Adresse    *)
    fcb[0]:=chr(ord(LW)-ord('A')+1)  (* Einstellen des Laufwerkes *)
    fcb[12]:= #0;
    a:= bdos($11,addr(fcb)) ;     (* Suchen nach erster Eintragung *)
    while a<255 do
      begin
        Eine_Eintragung(a);
        a:= bdos($12,addr(fcb)) ; (* Suchen weiterer Eintragungen *)
      end;
    HoleDirectory:=zaehler-1 ;
  end;
procedure DirWrite(Anzahl:byte);
  var n: byte;
  begin
    for n:=1 to Anzahl do  write(n:3,': ',DateiNamen[n],' | ');
  end;
procedure Dir(VerzMaske:string14);
  var i:byte;LW:char;mask:string11;
  begin
    while pos('*',VerzMaske)<>0 do delete(VerzMaske,pos('*',VerzMaske),1);
    if (VerzMaske[2]=':')and(length(VerzMaske)>1) then
      LW:=upcase(VerzMaske[1])                    (* Laufwerk angegeben  *)
    else
      begin
        LW:=char(bdos($19)+ord('A'));             (*  Standart-Laufwerk  *)
        VerzMaske:=lw+':'+VerzMaske
      end;
    if length(VerzMaske)=2 then mask:='???????????'   (* Keine Maske  *)
    else
      begin
        while (pos('.',VerzMaske)<11)and(pos('.',VerzMaske)<>0) do
          insert('?',VerzMaske,pos('.',VerzMaske));(* Auffuellen Dateinamen *)
        delete(VerzMaske,11,1);                    (* Loeschen des Punktes  *)
        while length(VerzMaske)<14 do
          VerzMaske:=VerzMaske+'?';                (* Auffuellen Extension  *)
        for i:=1 to 11 do mask:=mask+upcase(VerzMaske[i+2]);
      end;
    Datei_Anzahl:=HoleDirectory(LW,mask);
    DirWrite(Datei_Anzahl)
  end;
```

Bild 33 Eine große DIRectory-Variante

```
procedure ClearDirWrite4(Anzahl:byte);
  var x,y,n: byte;
  begin
    clrscr;                                        (* Loeschen Bildschirm *)
    for n:=1 to Anzahl do                 (* 4 Eintragungen auf einer Zeile *)
      begin
        x:=(n-1) mod 4*19+1;                      (* 4 x 19 Zeichen je Zeile *)
        y:=(n-1) div 4+4;                              (* erste Zeile  = 4 *)
        gotoxy(x,y);
        write(n:3,':',DateiNamen[n],' |');            (* Mit Numerierung *)
      end;
    writeln
  end;
procedure DirWrite5(Anzahl:byte);
  var n: byte;
  begin
    for n:=1 to Anzahl do
      write(DateiNamen[n],' |   ');   (* 5 Eintragungen auf einer Zeile *)
    writeln
  end;
procedure DirWrite6(Anzahl:byte);
  var n: byte;
  begin
    for n:=1 to Anzahl do
      begin
        write(DateiNamen[n],'  ');     (* 6 Eintragungen auf einer Zeile *)
        if n mod 6 = 0 then writeln
      end;
    writeln
  end;
```

Bild 34 Weitere Druckvarianten eines Verzeichnisses

```
type PathName = string[80];
     DirName  = string[12];
var  FCB  : Array[-7..35] of char;
     Regs : record case boolean of
              true :(ax,bx,cx,dx,bp,si,di,ds,es,Flags :integer);
              false:(al,ah,bl,bh,cl,ch,dl,dh            :byte    )
            end;
     DMA  : record
              FCB       :array [0..$14] of char: (* nur intern *)
              Attribut :byte;
              Zeit,Datum,SizeLo,SizeHi:integer;
              Filename :array[1..13] of char
            end;

function ErsterEintrag(Name:Pathname;Attr:byte):Dirname;
  begin
    with Regs do begin
      Name:=Name+#0 ;      ah:=$1A;
      ds  :=Seg(DMA);      dx:=Ofs(DMA);
      MsDOS(Regs);
      ah  :=$4E;           cx:=Attr;
      ds  :=Seg(Name[1]); dx:=Ofs(Name[1]);
      MsDOS(Regs);
      if odd(Flags) then ErsterEintrag:=''
        else with DMA do
          ErsterEintrag:=copy(Filename,1,pred(pos(#0,Filename)))
  end end;
function NaechsterEintrag:Dirname;
  begin
    with Regs do begin
      ah:=$4F;
      MsDOS(Regs);
      if odd(Flags) then ErsterEintrag:=''
        else with DMA do
          ErsterEintrag:=copy(Filename,1,pred(pos(#0,Filename)))
  end end;
Procedure dir(name:Pathname);
  var n:Pathname;
  begin
    n:=erstereintrag(name,0);
    while n<>'' do begin
      write(n,' ':16-length(n));
      n:=NaechsterEintrag
  end end;
```

Bild 35 DIRectory unter MS-DOS

dern auch im Array DateiNamen gesammelt. Somit stehen sie einer weiteren Auswertung zur Verfügung. Dateien, die die Attribute R/O und SYS besitzen, werden diesmal überlesen. Weiter wurde die Ausgabe auf den Bildschirm vom Lesen des Verzeichnisses selbst getrennt. Dadurch begünstigt ist es nun leicht möglich, das Ausgabeformat des Verzeichnisses zu ändern. Die Prozedur DirWrite, mit der im obigen Beispiel die Ausgabe auf dem Bildschirm vorgenommen wird, gestattet auch eine Durchnumerierung der aufgelisteten Dateien und zeigt jeweils vier Dateien in einer Zeile an. In Bild 34 werden noch drei weitere Verzeichnis-Ausgabe-Prozeduren angeboten, die jeweils ein anderes Format auf dem Bildschirm erzeugen.

Weder die Prozedur MiniDir, noch die Prozedur DIR, werden sofort und ohne Änderungen Aufnahme in Programme finden, da die Ansprüche an die speziell zu erbringende Leistung doch sehr unterschiedlich sein können; mit kleinen Änderungen werden sie aber sicher vielen Anforderungen gerecht.

Da die Auswertung eines Diskettenverzeichnisses natürlich nicht nur unter CP/M-kompatiblen Betriebssystemen interessant ist, wird in Bild 35 noch eine entsprechende Lösung für die Arbeit unter MS-DOS gezeigt. Da die prinzipielle Wirkungsweise den Lösungen unter CP/M entspricht, erübrigt sich eine nähere Beschreibung. Zu erwähnen ist lediglich, daß der Name, der der Prozedur dir übergeben wird, neben der Dateimaske sowohl Laufwerks- als auch Pfadangaben enthalten darf.

wird fortgesetzt

Fortsetzung von S. 302

Worte pro Bereich /8/ oder nur ganze Paragraphen /7/. Falls kleine Objekte in großer Zahl benötigt werden, kann dieser Overhead durchaus störend wirken. Eine einfache Methode dem auszuweichen (ohne einen vollständigen neuen Modul programmieren zu müssen), besteht darin, ALLOCATE und DEALLOCATE nicht direkt von Storage zu importieren, sondern einen eigenen Modul dazwischenzuschalten, der kleine Objekte in speziellen Verweisketten verwaltet und bei Bedarf jeweils mehrere zusammen reserviert. Anforderungen größerer Bereiche werden einfach weitergeleitet. Welche kleinen Objekte in dieser Art behandelt werden, kann dem Aufgabengebiet angepaßt werden (Bild 1). Auf eine Rekombination wurde in diesem Fall verzichtet. Es ist zu beachten, daß ALLOCATE in Prozessen teilweise (z. B. /9/) lokal wirkt, das heißt, der Speicherbereich wird dem des Prozesses entnommen.

Andere Module

Im Gegensatz zu Pascal wurde bei Modula-2 bewußt darauf verzichtet, Anweisungen zur Ein- und Ausgabe in die Sprache aufzunehmen. Die daraus resultierende größere Flexibilität schränkt aber andererseits die Portabilität ein, wenn Programme auf Standardkonfigurationen in Betriebssystemumgebungen laufen sollen. Aus diesem Grund hat Wirth in /1/ einen Bibliotheksmodul InOut als Quasi-Standard vorgeschlagen, der nahezu den in Pascal über Input/Output definierten Funktionsumfang aufweist. Für verschiedene Anwendungen ist die Beschränkung auf je einen Eingabe- und Ausgabekanal aber zu restriktiv. Außerdem differieren die Schnittstellen im Hinblick auf die zu übergebenden Parameter. Weitgehend portable Programme lassen sich daher nur entwickeln, wenn eine den konkreten Anforderungen entsprechende eigene Schnittstelle definiert wird, die sich dann natürlich auf vorhandene Module stützen kann, aber damit im allgemeinen auch systemabhängig wird.

Diese Tatsache sollte im Sinne einer sauberen Software-Technologie bei allen Modulen berücksichtigt werden, die für spezielle Anwendungen entwickelt wurden und nicht als Quelltext verfügbar bzw. systemspezifisch sind. Der sich aus diesen zusätzlichen Strukturierungen ergebende Overhead ist gering im Vergleich zu den daraus erzielbaren Vorteilen. Durch Verwendung von Prozedurvariablen oder durch Re-Export importierte Objekte (Aliasing z. B. in /7/) ist die erforderliche Effizienz in der Regel problemlos erreichbar.

Weitere Verbesserungen der Compiler und Linker sind zu erwarten, beispielsweise Inline-Abarbeitung einmalig gerufener Prozeduren. Selektives Binden, das heißt Einschluß nur der wirklich benötigten Prozeduren (vorteilhaft bei Verwendung multivalent nutzbarer Module) erlaubt bislang nur das JPI-System.

Aufmerksamkeit verdient weiterhin die Arbeit zur Standardisierung von Modula-2. Die vorliegenden Informationen lassen befürchten, daß der Standard in einigen Punkten erheblich von den bisher verbreiteten Sprachversionen abweichen wird. Das ist unerfreulich und verschlechtert die Chancen, daß sich ein Standard-Modula-2 durchsetzen wird. Ohnehin muß damit gerechnet werden, daß bereits vorliegende leistungsfähige Compiler wie /7/ und /9/ einen eigenen De-facto-Standard schaffen werden, ehe die Arbeiten der ISO abgeschlossen sein werden, und auch unabhängig davon, wie es ja bei Turbo-Pascal zu beobachten ist.

Literatur

/1/ Hauptmann, S.: Modula-2: Pascal ohne die Nachteile von Pascal. Mikroprozessortechnik, 2 (1988)10, S. 296
/2/ Wirth, N.: Programming in Modula-2. Springer-Verlag, Berlin, Heidelberg, New York 1982
/3/ Lampe, J.: Automatische Generierung von Fachsprachprozessoren mit ereignisgesteuerter Attributbewertung. INFO 88, Bd. 1, S. 134
/4/ Wirth, N.: Programmierung in Modula-2. Springer-Verlag, Berlin, Heidelberg, New York 1983
/5/ Cornelius, B.J.: Problems with the Language Modula-2. Software-Practice and Experience, 18(1988)6, S. 529
/6/ Cornelius, B.J.: Problems with the Report on Modula-2. Version 8, IST/5/13 Working Group Paper N103, BSI 1986
/7/ JPI-Modula-2, Jensen & Partners International, Mountain View 1987
/8/ VEB Robotron-Projekt: Modula-2 im BS DCP. Dresden 1987
/9/ Logitech Corp.: Modula-2/86, Redwood City 1986
/10/ Wirth, N.: Programmierung in Modula-2. Springer-Verlag. Berlin, Heidelberg, New York 1985
/11/ Wirth, N.: Revisions and amendments to Modula-2. Journal of Pascal, Ada and Modula-2, 4(1985)1, S. 25
/12/ Geissmann, L.: Overview of the Modula-2 Compiler M2RT11. Insitut für Informatik der ETH Zürich 1981

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Mathematik, Wissenschaftsbereich Mathematische Kybernetik und Rechentechnik, Mommsenstraße 13, Dresden, 8027; Tel. 4 63 40 82

Redabas-Tip

Command Master

Andreas Holländer

Bei der Arbeit mit dem Datenbanksystem Redabas können Kommandos eine recht beachtliche Länge erreichen. Enthalten sie Auswahlkriterien und/oder Feldnamen (copy, list usw.), sind Längen von 80 Zeichen und mehr keine Seltenheit. Besonders dann, wenn solche Kommandos mit geringen Änderungen mehrmals hintereinander genutzt werden sollen, die Erstellung eines Programmes aber nicht lohnt, ist das im folgenden vorgestellte Hilfsmittel nützlich.

Das Programm Command Master (CM.PRG) gestattet die Manipulation mit bis zu fünf Kommandos. In jedem dieser fünf Speicher stehen 80 Zeichen (eine Bildschirmzeile) für ein Kommando zur Verfügung, die bis auf 240 Zeichen aufgestockt werden können. Gestartet wird das Programm CM.PRG aus der Kommandoebene von Redabas (DO CM) oder direkt beim Redabasaufruf (REDABAS CM). Nach dem Start und nach der Abarbeitung einer Operation des Command Masters erscheint auf der untersten Bildschirmzeile die Ausschrift **CM : 1 >**, die die Nummer des Kommandospeichers enthält, der sich momentan im Zugriff befindet.

Es sind folgende Operationen realisierbar:

i	– Eingabe und sofortige Ausführung eines Kommandos. Dieses Kommando steht dann zur nochmaligen Ausführung oder zur Korrektur zur Verfügung.
c	– Korrektur des Kommandospeichers
e	– Ausführung des im aktuellen Speicher befindlichen Kommandos
d	– Kopieren des aktuellen Speicherinhaltes in einen anderen Kommandospeicher
s	– Auslagern der Inhalte der Kommandospeicher auf Diskette
l	– Laden der Kommandospeicher von Diskette
+	– Vergrößerung des Kommandospeichers um 80 Zeichen
1 ... 5	– Änderung des Speicherzugriffs
?	– Anzeige der Operationsliste des Command Master
q	– Rückkehr zur Kommandoebene von Redabas

Mit diesem einfachen Hilfsmittel kann der mit den Bedienungsgrundlagen des Datenbanksystems vertraute Nutzer seine Arbeit deutlich effektivieren. Die Lücke zwischen der mitunter schreibaufwendigen direkten Bedienung des Systems und der in der Erarbeitung intelligenzintensiven Programmsteuerung kann geschlossen werden.

```
**** cm.prg ** command master v.880906 ****
set talk off
set colon off
erase
store '                                        ' to cmds
store cmds+cmds to cmds
if type(cmd1)='U'
   release all except cmds
   store cmds to cmd1
else
   release all except cmd?
endif
store '1' to cmdn
store ' ' to goon
@ 10,19 say '##########  COMMAND MASTER ##########'
do while goon#'Q'
clear gets
?
@ 25,0 say 'CM : '+cmdn+'> ' get goon picture '!'
read
do case
   case goon='I'
        accept '?' to cmd&cmdn
        ?
        set talk on
        store cmd&cmdn to ex
        &ex
        set talk off
        ?
   case goon='C'
        clear gets
        @ 1,0 get cmd&cmdn
        read
   case goon='E'
        ?
        set talk on
        store cmd&cmdn to ex
        &ex
        set talk off
        ?
   case goon='D'
        accept 'nach ?' to ncmdn
        if ncmdn>'0'.and.ncmdn<'6'
           store cmd&cmdn to cmd&ncmdn
           store ncmdn to cmdn
        else
           ? '*** ? - Hilfe ***'
        endif
   case goon='S'
        save to cm.var
   case goon='L'
        if file('cm.var')
           restore from cm.var
        else
           ? '*** keine Datei CM.VAR ***'
        endif
   case goon='+'
        if len(cmd&cmdn)<161
           store cmd&cmdn+cmds to cmd&cmdn
        endif
   case goon>'0'.and.goon<'6'
        store goon to cmdn
        if type(cmd&cmdn)='U'
           store cmds to cmd&cmdn
        endif
   case goon='?'
        ?
        ? '##### ###### COMMAND MASTER ##### #####'
        ? 'i          - Eingabe und Ausfuehrung'
        ? 'c          - Korrektur'
        ? 'e          - Ausfuehrung'
        ? 'd          - Kopieren'
        ? 's          - Auslagern auf Diskette'
        ? 'l          - Laden von Diskette'
        ? '+          - Vergroesserung'
        ? '1...5        Speichernummer'
        ? '?            Anzeige der Operationsliste'
        ? 'q            Ende'
        ? '#######################################'
        ?
   case goon='Q'
   otherwise
        ? '*** ?   Hilfe ***'
endcase
enddo
? '########################################## Ende ###'
set talk on
set colon on
return
```

Der Apple Macintosh

Dr. Hans Dolleschel, Berlin

Vor etwa fünf Jahren wurde von der Firma Apple Computer die Personalcomputerfamilie Macintosh vorgestellt und damit die PC-Entwicklung auf der Grundlage von Motorola-Prozessoren fortgesetzt. Bekanntlich kann Apple für sich in Anspruch nehmen, mit dem Apple II schon 1977 die damals neue Klasse der „persönlichen Computer" mit begründet zu haben. Jedoch gelang es der Firma dann auf Dauer nicht, sich der Übermacht des Computergiganten IBM zu erwehren, der 1981 seinen PC auf Basis des Intel-Prozessors 8088 etablierte. Die massenhafte Produktion dieses Typs und seiner Nachfolger und nicht zuletzt die noch größere Verbreitung durch mehr oder weniger kompatible PCs anderer Hersteller führten letztlich dazu, daß PCs mit Intel-Prozessoren unter dem Betriebssystem PC-DOS bzw. MS-DOS quasi zum Industriestandard wurden. Heute kann wohl kein Computerproduzent in der Welt diese Tatsache unberücksichtigt lassen. Dennoch – oder gerade deshalb – scheint es zweckmäßig, den Blick auch einmal auf Entwicklungsrichtungen zu werfen, die man durch Gewöhnung an die verbreiteten PCs schnell einmal übersieht. Dabei wird man unter Umständen feststellen, daß es einige weniger stark verbreitete Mikrorechnersysteme gibt, die manche heutige Neuerungen der MS-DOS- oder auch der OS/2-Umgebung schon seit Jahren und in teilweise besserer Qualität aufweisen. Hier seien nur die Windowtechnik mit grafischer Benutzeroberfläche einschließlich Mausführung, verschiedenste pixel- und objektorientierte Grafikprogramme und die gegenwärtig modern gewordenen DTP-Programme angeführt.

Das Macintosh-Konzept

Zunächst einmal fällt rein äußerlich die eigenwillige, kompakte, hochformatige Bauform des Mac Plus und der beiden Mac SE auf (vergleiche Bild 1, das auch die Transportfreundlichkeit erkennen läßt), die dann mit den Mac II/IIx (siehe Bild 2) und erst recht mit dem Mac IIcx (Bild 3) verlassen wurde, um mit dem Mac SE/30 (Bild 4) wieder „alte" Akzente zu setzen. Die äußere Form ist jedoch zweitrangig gegenüber einem Vorzug der Computerfamilie, der sich in sehr kurzer Zeit jedem ungeübten Anwender erschließt: Das ist die mit *Human Interface* umschriebene „menschliche" Benutzeroberfläche, die man zudem noch an eigene Bedürfnisse und Gewohnheiten anpassen kann und die dennoch einem für die DOS-Welt beispiellos einheitliche Benutzerinterface für die Grundfunktionen folgt. Und „menschlich" ist auch eine erstaunliche Toleranz gegenüber Bedienfehlern, die ein leichtes Rücknehmen gestattet und nicht gleich zu Datenverlusten, Arbeitszeitverlusten oder häufigen Systemabstürzen führt. Im Gegensatz zum Betriebssystem PC-DOS bzw. MS-DOS, einschließlich dessen historischem Vorgänger CP/M, bei denen die Kenntnis wesentlicher Betriebssystembefehle unerläßlich für die Bedienung der PCs ist, wird in der Macintosh-Welt konsequent mit einer einheitlichen Grafikoberfläche gearbeitet. Ohne Befehle auswendig lernen zu müssen und ohne Codes oder komplizierte Prozeduren kann sich der Benutzer allein mit der Maus und den Icons (oder deutsch Ikonen) genannten Symbolbildchen so bewegen, wie er es von seinem mit Dokumenten belegten Schreibtisch her gewöhnt ist.

Macintosh – geschichtlich gesehen

Die eigentliche Grundidee einer leicht bedienbaren Benutzeroberfläche durch *bit-mapped display* geht zurück auf mehrere Entwicklungen verschiedener Wissenschaftler und Laboratorien ab etwa 1971 – lange bevor es eine Firma Apple gab. Vom Stanford Research Institute wurde die Idee der Maus als Eingabemedium geboren, und Rank Xerox nahm mit einigen Wissenschaftlern von dort auch diese und weitere Ideen vom „Einklicken" in Bildschirmfenster auf. Bei Xerox wurden dann mit Entwicklungen zum allerersten „persönlichen" Computer überhaupt, dem Alto Computer, auch die ersten Schritte in eine neue Richtung getan, die jedoch kommerziell niemals zum Tragen kamen. Nebenbei bemerkt wurde hier bereits die Idee des WYSIWYG-Prinzips geboren (What you see is what you get – frei übersetzt etwa: So wie du es auf dem Bildschirm siehst, so bekommst du es auch ausgedruckt.), und dieses Konzept betrifft beim Macintosh durchaus nicht nur die Ausdrucke von Dokumenten, Zeichnungen oder Layouts.

Es gab weitere Entwicklungen in Richtung Desktop-Benutzeroberfläche, aber die beiden tragenden Entwickler verließen die Firma Xerox 1981 in Richtung Atari und Apple und das neue Computer-Produkt Xerox Star fand zwar Beachtung, scheiterte aber kommerziell.

Bei Apple wurde dann, parallel zu den berühmten Apple I und II, von den Apple-Gründern Steve Wozniak und Steven Jobs die Entwicklung zur Lisa aufgenommen, die auf einigen Erkenntnissen von Star und weiteren Entwicklungen aufbaute und mit fundamentalen Neuerungen einschließlich sieben integrierten Anwenderprogrammen aus der „Schreibtischumgebung" schließlich 1983 vermarktet wurde. Allein, der große Erfolg blieb immer noch aus, denn die nun vielbeachtete Lisa war immer noch zu teuer für die Zielgruppe, für die sie eigentlich bestimmt war. Aber kurz nach Aufnahme der Entwicklungen zur Lisa brachte Steven Jobs fundamentale Lisa-Ideen (insbesondere des grafischen Umfeldes) in eine bereits bestehende andere Entwicklungsgruppe bei Apple ein: Der „Appliance Computer" sollte mit dem Ziel 70 % Lisas Leistung bei 20 % der Kosten entwickelt werden /1/. Mit der Einbindung der von dem talentierten Softwareentwickler Bill Atkinson geschaffenen QuickDraw-Routinen und einem hochmotivierten Team gelang dann dem später Macintosh genannten Projekt der Durchbruch. Es ist Apples Verdienst, mit dem Macintosh den ersten massenproduzierten PC der „Desktop"-Technologie geschaffen und kommerziell zum Erfolg geführt zu haben. Natürlich erfuhr der heute verwendete Mac anschließend noch einige wichtige Verbesserungen, aber das erste Modell, damals noch mit nur 128 KByte RAM und den einseitigen Disketten mit 400 KByte, begründete die 32-Bit-Familie von Apple auf der Basis der Motorola-Prozessoren der 680x0-Serie. Am



***Bild 1** Der Macintosh Plus ist der älteste und am weitesten verbreitete Vertreter der Mac-Familie. Äußerlich ist er den später hinzugekommenen SE-Typen mit höherem Bauelemente-Integrationsgrad und einigen Verbesserungen noch ähnlich. Über Jahre hinweg hat er seine Bedeutung beibehalten und wird auch heute noch in hohen Stückzahlen produziert, vorwiegend als preiswerter Mac für den privaten Bereich oder Einsätze in Kleinbetrieben.*

24. Januar 1984 präsentierte Apple mit typisch amerikanischem Pomp und einem berüchtigten Fernsehspot der Öffentlichkeit den ersten Macintosh (mehr dazu in /2/).

Icons – Eintritt in eine bildhafte Welt

Nach diesem Ausflug in die Historie des Macintosh nun etwas ausführlicher zu den wesentlichen Merkmalen dieser Computerfamilie. Wie bereits eingangs erwähnt, ist es für den Mac charakteristisch, daß der Benutzer seinen Schreibtisch als Darstellung auf dem Bildschirm wiederfindet – einschließlich der einzelnen Utensilien. Da gibt es auf dem „elektronischen Schreibtisch" Schnellhefter (englisch Folder), leere Blätter, Bleistift, Radiergummi, Pinsel und je nach Programm Hunderte weitere handliche Werkzeuge, in die der Pfeil der Maus sich wandeln läßt. Man muß nur mit der Maus auf das Objekt tippen, welches man irgendwie verändern will, und schon wird der gewünschte Vorgang ausgelöst. Etwas exakter ausgedrückt: Zunächst wird per Mausbewegung und -druck ein Objekt ausgewählt, und dann wird aus dem Menü ein Befehl ausgewählt, mit dem man die dazu gewünschte Veränderung angibt. Oder aber es wird ein bestimmtes Objekt mit Hilfe der Maus auf ein anderes gelegt und allein damit eine Operation ausgelöst (z. B. wird so ein Dokument in einen Aktendeckel einsortiert – einschließlich des selbständigen Anlegens eines Verzeichnisses usw.). Es stellt sich heraus, daß selbst diese Beschreibung hier aufwendiger ist als ein erster unbefangener Versuch mit diesen selbsterklärenden Instrumenten. Und damit überzeugt ein weiterer Vorteil, der in der Literatur gelegentlich mit Konsistenz bezeichnet wird: In dieser Macintosh-Welt mit ihrer einheitlichen Oberfläche kann man nahezu jedes von beliebigen Softwarehäusern geschriebene Programm sofort benutzen, ohne es eigentlich genau zu kennen und vor allem, ohne eine Dokumentation oder ein Handbuch lesen zu müssen. Nicht wenige Macintosh-Benutzer arbeiten mit Programmen, ohne jemals dazu eine Erläuterung gelesen zu haben. Bei komplexen integrierten Softwarepaketen (z. B. Works oder Excel) oder hoch spezialisierter Software (z. B. Bildverarbeitungsprogrammen) kann man natürlich nicht mehr alle Raffinessen voll ausnutzen, wenn man auf

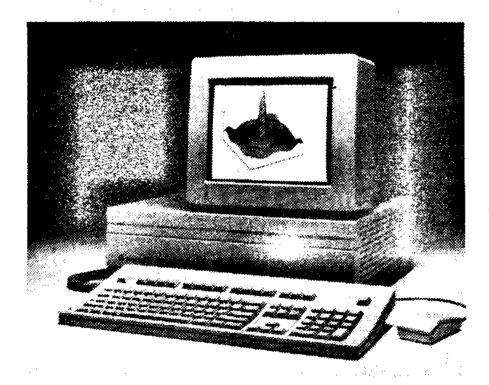


Bild 2 Der Macintosh II erinnert mit seinem abgesetzten Bildschirm schon eher an die gewohnten PCs. Er ist das 1987 eingeführte Grundmodell für die überwiegend professionellen Anwender in Industrie, Handel, an Bildungseinrichtungen und Universitäten und vor allem in Büros.

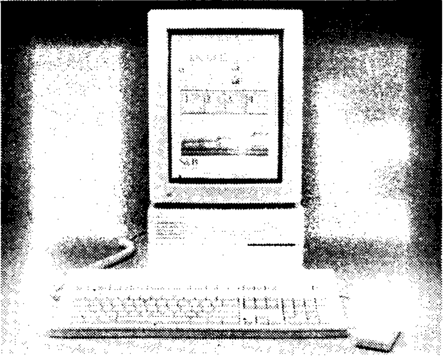


Bild 3 Den Macintosh IIcx als jüngstes Modell könnte man besser mit Workstations vergleichen als mit Hochleistungs-PCs. Sein 68030-Mikroprozessor, mathematischer Koprozessor und bis zu 8 MByte Hauptspeicher (sowie bis 160 MByte Festplatte) prädestinieren ihn für rechenintensive wissenschaftlich-technische Aufgaben, anspruchsvolle DTP-Anwendungen mit Bild- und Grafikverarbeitungen, CAD-Anwendungen und Projektplanungen, und für hervorragende Kommunikationsaufgaben in unterschiedlichsten Rechnernetzen (besonders auch als Server). Der Mac IIcx hat ein neuartiges modulares Gehäusekonzept, das auch einen senkrechten (Tower-)Betrieb zuläßt und besonders servicefreundlich ist.

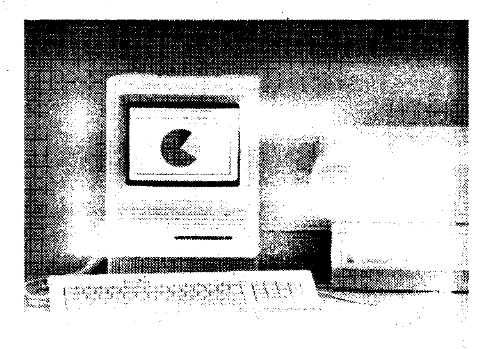


Bild 4 Der Macintosh SE/30 wird von Apple als „Flaggschiff" bezeichnet und ist im Gegensatz zum kurz darauf erschienenen individuell konfigurierbaren modularen Mac IIcx wieder in der bewährten Kompaktform der Mac Plus und Mac SE ausgeführt. Er hat gemeinsam mit dem IIcx mit dem 68030 die schnellste CPU der Macintoshs. Auf den 32-Bit-Daten- und -Adreßbus kann mit einem neuen Steckplatz direkt zugegriffen werden. Der SE/30 wird mit eingebauter 40-MByte-Festplatte (3,5 Zoll) und 2 oder 4 MByte Hauptspeicher ausgeliefert.

Bedienerliteratur verzichtet. Aber man denke nur an die Berge von Literatur, die ein normales MS-DOS-Programm begleiten und deren Studium kaum zu umgehen ist.

An einem Icon soll einmal der Vergleich zur Arbeit mit DOS-Befehlen deutlicher werden: Für die „Vernichtung" von Dokumenten, Ordnern oder auch ganzen Programmen steht auf der „Schreibtischplatte" (desk top) ein Papierkorb bereit. Dorthinein kann man mit der Maus alles Überflüssige „ziehen". Im Gegensatz zu Daten fragt der Mac bei Programmen noch mal nach, ob man das wirklich wegwerfen will, und selbst nach der Bestätigung bleibt alles noch so lange im Papierkorb liegen und kann jederzeit wieder hervorgeholt werden, wie man genug Speicherplatz hat oder man keine andere Diskette einlegt. Wenn der RAM-Platz allerdings von anderen Programmen benötigt wird oder aber wenn man einen entsprechenden Menübefehl aus der Menüleiste auswählt, dann wird der Inhalt des Papierkorbes „geleert", das heißt, das Dokument oder das Programm wird tatsächlich (und nicht nur symbolisch) gelöscht – und zwar dort, von wo man es herholte: auf der Diskette, der Festplatte oder im Hauptspeicher. Gleichzeitig wird der nun frei gewordene Speicherplatz angezeigt und steht zur Verfügung. Dieser gesamte Vorgang läuft also ohne einen einzigen Befehl ab und ohne auch nur ein einziges Mal die Tastatur berühren zu müssen.

Fensterln

Noch ein Wort zu den Macintosh-eigenen Fenstern, den Windows, wie sie in Ansätzen in GEM oder mit Microsoft Windows langsam in die DOS-Welt eingeführt werden (Microsoft hatte ebenso wie Apple an den Routinen zu den Windows schon vor 1984 gearbeitet und von Apple eine Lizenz erhalten). Auf allen Macintoshs können fast beliebig viele Fenster neben-, auf- unter- oder übereinander geöffnet werden. Das zuletzt bearbeitete ist das aktive, das heißt, alle Aktionen beziehen sich darauf. Sofern man dieses aber mit dem mausgesteuerten Zeiger (Pfeil oder beliebiges Sinnbild) verläßt, wird ein neues Fenster augenblicklich aktiv, wenn man es mit einem Mausklick an beliebiger Stelle berührt. Man kann jedes Fenster beliebig vergrößern und an jede Stelle des Bildschirmes „kleben". Alle diese Fensterparameter werden abgespeichert und bleiben auch erhalten, wenn das Programm verlassen wird. Wer also auf seinem Schreibtisch eine schöpferische Unordnung für seine Gedanken braucht – bitte schön, auch das geht. Und wenn dann im Ernstfall doch mal etwas gesucht werden sollte, dann hilft ein geeigneter Mausklick (und entsprechende Software) sicher weiter. Solch kleine Zusatzsoftware (Desk Accessories oder Werkzeuge genannt – z. B. Auskunfts- und Suchprogramme, Taschenrechner, Notizbücher, Terminkalender, Sprachausgaben, Zeichenprogramme usw.) ist jederzeit, also auch aus geöffneten Anwenderprogrammen heraus, vom Menü aus aufrufbar. Es gibt Hunderte dieser Programme, und da man sie sich ganz nach Bedarf selbst ins Menü installieren kann (auch das ist denkbar einfach), läßt sich auch damit auf einfache Weise die Benutzerschnittstelle des Macintosh den persönlichen Bedürfnissen anpassen – und nicht umgekehrt!

Bekanntermaßen beginnt sich heute auch in der DOS-Welt mit dem Betriebssystemzusatz Windows von Microsoft und in naher Zukunft mit der Auslieferung des Presentation Managers für die IBM-Produktlinie PS/2 eine Hinwendung zur grafisch orientierten Benutzeroberfläche zu vollziehen, wie sie beim Apple Macintosh (und, mit geringerer Bedeutung, auch beim Atari 1040 ST und beim Commodore Amiga) schon längst verwirklicht wurde. Die Windows-Lösung und erst recht der noch umstrittene Presentation Manager stellen jedoch im Gegensatz zu den vom Motorola-Mikroprozessor bereitgestellten Fähigkeiten eine nachträglich künstlich „übergestülpte" Hülle für ein an sich befehlsorientiertes Betriebssystem dar. Das führt neben einer Verlangsamung zu ziemlichem Speicherplatzbedarf. Immerhin braucht OS/2 wenigstens 2 MByte Hauptspeicher, um überhaupt vernünftig arbeiten zu können, und auch für den multitaskingfähigen Zusatz Windows zum DOS empfiehlt der Hersteller mindestens 2 MByte RAM, besser seien jedoch 4 MByte. Das heißt, die DOS-Grenze von 640 KByte für die Hauptspeicherverwaltung mußte schon aus diesem Grunde aufgelöst werden.

Kompatibilität und Standards

Der erwähnte eigenständige Weg Apples mit der Macintosh-Familie führte nicht zur Abschottung von anderen Betriebssystemen. Nach einem firmeninternen, außerordentlich wechselhaften Abklärungs- und Marktanpassungsprozeß (der Apple beinahe an den Rand des Ruins gebracht hatte) werden heute für die neuen Macintoshs neben dem herstellereigenen noch viele unterschiedlichste Betriebssysteme (z. T. sogar gleichzeitig laufend) angeboten. Insbesondere die Entwicklung zur IBM-Kompatibilität ohne Aufgabe der eigenen Vorzüge hat nicht unwesentlich zu dem heutigen Erfolg der Serie bei professionellen Anwendungen beigetragen. Dafür im weiteren einige Beispiele.

Apple reagierte – natürlich wie andere auch – auf die Voraussagen von Marktforschungen, daß ab den 90er Jahren insbesondere für wissenschaftlich-technische Anwendungen in etwa 90 Prozent aller Fälle das Unix-Betriebssystem eingesetzt werden wird. Denn immerhin sind die Hochschulen und die wissenschaftlichen Einrichtungen ein Haupteinsatzgebiet der Macintoshs. Ab dem Macintosh II gibt es das Betriebssystem A/UX, welches kompatibel zum Unix-Release V und Berkeley 4.2 ist, das heißt, es sind nahezu alle wichtigen Unix-Programme lauffähig, und zwar in einem Macintosh-typischen Fenster mit Untermenüs, aus denen die erforderlichen Betriebssystemfunktionen aufgerufen werden können. Das soll helfen, die öfters kritisierte Unix-Bedienerschnittstelle zu verbessern. Außerdem werden zusätzliche Leistungen gegenüber „normaler" Unix-Umgebung bereitgestellt wie *Auto Recovery Routine* nach einem Systemabsturz, *Auto Configuration* und Zugriff auf eine umfangreiche Mac Toolbox Library.

Bei allen Macintosh-Typen bestand schon längere Zeit die Möglichkeit eines problemlosen Daten- bzw. Fileaustausches zwischen DOS-PC und Mac, genauso übrigens wie zwischen völlig fremden PC-Systemen, zum Beispiel einem japanischen NEC-PC und dem Mac. Dabei werden entweder von PC zu PC über eine serielle Schnittstelle oder mit DOS-Disketten über ein spezielles externes 5,25-Zoll-Laufwerk für die entsprechenden DOS-Formate und mit integrierter Konvertierungshardware/-software Daten (Files) ausgetauscht. Ab dem Erscheinen des Macintosh II und rückwirkend auch bei den Mac SE können nun im Gegensatz zum einfachen ge-

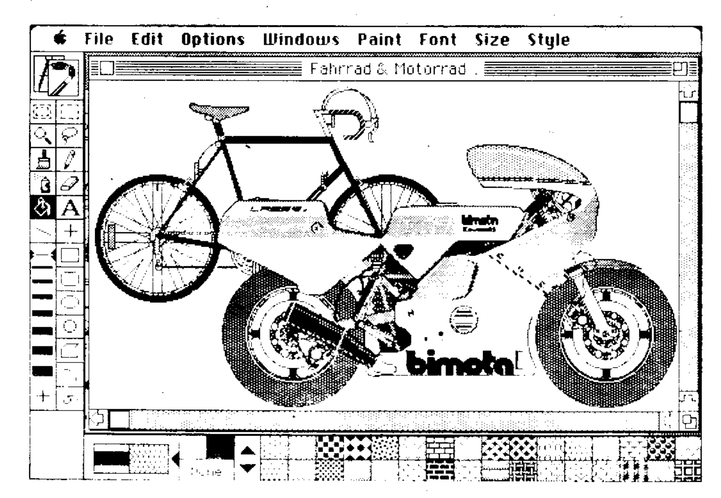




***Bild 5** Darstellung einer pixelorientierten (bitmapped) Grafik in einem einfachen Malprogramm. Das Bild wurde nur mit Maus und den in den Menüleisten gezeigten Werkzeugen „gemalt". Die untere Menüleiste zeigt eine Auswahl von gerade zugreifbaren Mustern (bzw. Farben bei mehrfarbfähigen Macs).*

***Bild 6** Im Gegensatz zu Bild 5 hier die Darstellung eines pixelorientierten Bildes, das aus einem gescannten Halbtonbild mit einem umfangreichen Bildbearbeitungsprogramm (jeder Bildpunkt hat eine 4-bit, 6-bit, 8-bit oder noch höhere Graustufenlösung) gewonnen wurde und erst dann (zur druckfähigen Darstellung) in ein Rasterbild mit reinen schwarz-weißen Punkten umgewandelt wurde. Die hochauflösenden Bilder mit Graustufen sind für Qualitätsdrucke über eine Lichtsatzmaschine erforderlich, wie sie anspruchsvolles DTP benötigt.*

genseitigen „Verstehen" auch die meisten MS-DOS-Programme direkt im Mac abgearbeitet werden. Das wird durch eine Koprozessorkarte erreicht, mit der MS-DOS emuliert wird. Von normaler Macintosh-Umgebung aus wird die originale MS-DOS-5,25-Zoll-Diskette gestartet, und die übliche DOS-Umgebung erscheint innerhalb eines Mac-Fensters. Mit dem neuen Multifinder (einer Art Multitasking-Möglichkeit des Macintosh-betriebssystems) kann ein Mac-Benutzer jetzt ohne jeden Zeitverlust (d. h. ohne ständiges Sichern, Schließen, Neuöffnen) zwischen den beiden Systemen wechseln, Daten austauschen (ausschneiden, kopieren, einsetzen usw.) und gleichzeitig die zahlreichen Mac-typischen Zusatzprogramme (Desk Accessories) weiterhin benutzen. Und das Ganze läuft aufgrund der höheren Frequenz des Mac bis zu 25 Prozent schneller ab als auf den „eigentlichen" MS-DOS-Computern.

Wie in MP 2/89, S. 61, gemeldet, kann der Mac IIx neben diesen Prozessen durch das neue Diskettenlaufwerk FDHD neben allen Mac-eigenen Formaten auf 3,5-Zoll-Disketten (800 KByte/doppelseitig und 400 KByte/einseitig für die älteren sowie für das neue Format 1,44 MByte) nun auch die neuen 3,5-Zoll-Disketten des IBM OS/2-Systems und des MS-DOS-Formates lesen und beschreiben. Das ist Konsequenz im Sinne der erwähnten Marktanpassung.

Erwähnt sei noch, daß neben den Hardware-Zusatzkarten zunehmend auch Softwareentwicklungen von unabhängigen Computerfirmen auftauchen, die die Macintoshs neben ihrer normalen Funktion in vollständige XT- oder AT-Clones verwandeln können. Über den Vorteil solcher Lösungen, das nicht mehr so günstige Zeitverhalten und den Kostenaufwand läßt sich allerdings streiten.

Mit weiteren zusätzlichen Steckkarten erschließen sich den Macintoshs auch andere IBM-Verbindungen. Beispielsweise kann ein Mac die IBM 3278- oder IBM 3279-Terminals emulieren und so Host-Kommunikation durchführen – aber auch hier unter Beibehaltung seiner Mac-typischen Umgebung und Benutzung derer Zusatzprogramme. Dazu können dann vom Host Daten heruntergeladen, bearbeitet und zum Großrechner zurückgebracht werden.

Einen vielbeachteten Schritt Apples in Richtung „Connectivity" brachte die Anfang 1988 eingeschlagene Kooperation mit dem namhaften Computerhersteller Digital Equipment Corp. (DEC) mit dem Ziel, die Macintosh-Computer und AppleTalk-Netze mit den VAX-Rechnern verbinden und in DECnet-Netzwerke integrieren zu können sowie gemeinsame Produkte zu entwickeln. Die Macintoshs könnten dann vor allem als intelligente Terminals für die VMS-Betriebssystemumgebung von DEC eingesetzt werden – eine wichtige Verbindung für beide Hersteller, wenn man bedenkt, daß bereits zu diesem Zeitpunkt in den USA 36 Prozent aller VAX-Benutzer schon Apple Macintoshs eingesetzt hatten.

Diese hier unvollständig und ohne Prioritäten angeführten Beispiele unterstreichen die Absicht Apples, zunehmend auf Kompatibilität zu verbreiteten, aber Macintosh-fremden Betriebssystemen hinzuwirken. Das wird selbstverständlich in erster Linie mit Blick auf die Absatzchancen eigener Apple-Produkte getan, bringt aber natürlich für viele Benutzer verschiedener PC-Systeme oder gar Minicomputer oder Mainframes, und insbesondere auch für kleinere und mittelständische Betriebe sowie Hoch- und Fachschulen, große Vorteile. Man vergleiche damit die ursprüngliche Absicht von IBM, mit ihrer PS/2-Einführung lästige IBM-Clone-Hersteller abzuschütteln – und sei es auf Kosten der Aufgabe der Kontinuität und der Kompatibilität zur eigenen weltweit eingeführten Produktlinie PC-DOS bzw. MS-DOS.

Alle bisher betrachteten Wege und Lösungen zur Schaffung einer mehr oder weniger geeigneten Kompatibilität zwischen Anwendungen auf unterschiedlichen Betriebssystemen werden bei einer geeigneten Einbindung von unterschiedlichen PCs in lokale Datennetze (LAN) eigentlich überflüssig. Mit solcherart Verbindung unterschiedlicher PC-Architekturen in der betrieblichen EDV-Organisation – eine Aufgabe von LANs bei der Büroautomatisierung – ist es heute bereits insbesondere in den USA durchaus normal geworden, Macintoshs in LANs mit beispielsweise Ethernet-Konzepten einzubinden (hierfür werden Ethernet-Adapter eingebaut), wo sie dann je nach Netzsoftware mit Hostrechnern zusammenarbeiten, in Unix-(bzw. den für PCs gängigeren Xenix-Netzen) arbeiten oder mit DOS-Geräten in verschiedenartigsten Netzstrukturen kommunizieren und auf gemeinsame Datenbasen zugreifen. Die bekannten Novell-Netze sind in unterschiedlichsten Ausführungen auch für die Mac-Welt verfügbar und waren ein nicht zu übersehender Schwerpunkt auch auf der CeBit '89.

Vom harten Softwaremarkt

Einige weltweite Softwareklassiker haben bereits 1984 ganz wesentlich den Erfolg der Macintoshs eingeleitet (beispielsweise von Microsoft) Basic, File, MultiPlan, Chart, Word und insbesondere das einzigartige grafisch orientierte Tabellenkalkulationsprogramm Excel sowie Grafikprogramme wie MacPaint (pixelorientiert, z. B. Bilder) oder MacDraw (objektorientierte Zeichnungen) von Apple). Die Beliebtheit der Macs an den Hochschulen und Universitäten (anfangs vor allem der USA) und die Apple-Vermarktungspolitik, ihre nicht gerade billigen Geräte für Studenten oder Bildungseinrichtungen preisgünstiger zu verkaufen, führten zu einer wahren Flut kleinerer von Studenten und Fans entwickelter Utilities und kostenloser öffentlicher Public-Domaine-Software. Dieses Angebot trug seinerseits wiederum zweifellos zu einer schnelleren Verbreitung der Macs bei.

Seit etwa 1986 erklärten nahezu alle namhaften Softwarehäuser ihre Bereitschaft, für die Mac-Welt Programme zu entwickeln, und es setzte ein merkliches Gerangel auf dem Softwaremarkt ein. Neben Microsoft und Apple selbst trat Aldus mit PageMaker auf und Ashton-Tate u. a. mit dem relationalen Datenbanksystem dBase-Mac mit Kompatibilität zum erfolgreichen dBase III Plus und mit der Textverarbeitung FullWrite. AutoDesk kündigte die Version 10 von AutoCAD für den Macintosh an. Oracle, das vielleicht wichtigste Datenbanksystem der professionellen Mikrocomputerebene, ist nun auch für Macintosh verfügbar und beschleunigt das Vordringen in Minicomputer- und Workstationebenen. Die von Sun bereitgestellte Netzsoftware TOPS ist ein enormer Verkaufsschlager (mit Kompatibilität u. a. zu den Netzen 3Com3+, Novell S-Net, AT&T Starlan, IBM PC-Net, ARCNet, Corvus) und greift zudem auf das im Mac bereits integrierte AppleTalk zurück. Das bedeutet für einen Macintosh-



Bild 7 Layoutgestaltung als Beispiel in einem DTP-Programm

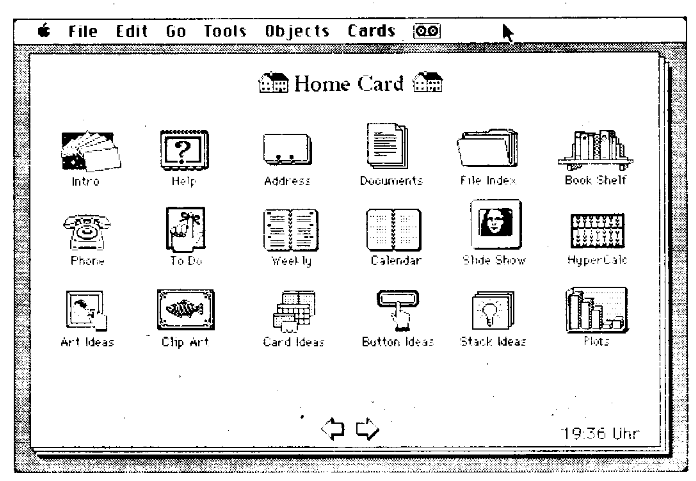


***Bild 8** Die „Homecard" des HyperCard, von der aus man in die häufiger benutzten Stacks oder „Stapel" ohne Umwege gelangen kann, indem man in eines der Ikonen klickt. Hierher gelangt man auch aus jeder HyperCard-Anwendung schnell zurück, wenn man nicht den direkten Weg zwischen allen Stapeln benutzen möchte.*

Besitzer einen durchaus wesentlichen Vorteil: Er braucht nicht wie bei anderen PCs Netzkarten und -software zusätzlich kaufen, da dieses ja bereits im Mac vorhanden ist. Der Platz in diesem als Übersicht gedachten Beitrag reicht nicht annähernd aus, um eine Vorstellung von der Vielfalt der vorhandenen Anwendungssoftware zu vermitteln.

Vorbildlicher Macintosh

Diese Überschrift darf man wörtlich nehmen – der zunehmend erfolgreiche Macintosh, der so sehr von der IBM-Linie abstach und so leicht bedienbar war, diente als Vorbild für die verschiedensten Modelle und Architekturen. Einige Tendenzen, die durch ihn in der DOS-Welt ausgelöst wurden, sind ja schon eingangs gestreift worden: Windows und Maus zusammen mit Menüs (insbesondere das Pull-down-Menü) haben Standards gesetzt, ohne die heute kein PC-System mehr angeboten werden kann. Die grafische bit-mapped-Oberfläche ermöglichte erstaunlich leicht bedienbare, hochauflösende Zeichen- und Malprogramme bis hin zum CAD; die vielfältigsten Mal- und Zeichenwerkzeuge (Bleistift, Radiergummi, Pinsel, Farbkanne zum Ausgießen, Spraydose usw.) können aus einem bildhaften Menü geholt werden und gestatten verblüffende Effekte selbst für Ungeübte (siehe als Beispiel Bild 5). Diese bereits auf den ersten Macs als „Grundausrüstung" vorhandenen Programme findet man heute in zahlreichen DOS-Nachahmungen. Nicht selten konnte man in der IBM-orientierten PC World (z.B. /3/) ab 1986 Formulierungen finden wie: „Like Apples MacPaint, the (xyz) uses icons to denote program functions". Oder juristische Abhandlungen, ob GEM, Windows usw. die Copyrights von Apple verletzen. Oder im PC Magazine /4/: „GEM Write looks a lot like MacWrite running on a Macintosh ..." oder „GEM Desktop, a Macintosh-like mouse and windowing environment ...". Es ging nicht mehr ohne diese Macintosh-Merkmale.

Aber nicht nur bezüglich DOS gab es Auswirkungen und Anregungen. Selbst die von Japan eigenständig entwickelte offene Computerarchitektur des TRON-Projektes, eine zweifellos richtungweisende und wichtige Entwicklung für 32-Bit-Computer und aufwärts, die die besonderen japanischen Bedingungen (u. a. Kanji-Schriftsprache) excellent berücksichtigt, hat Anleihen vom Macintosh aufzuweisen.

Bei der Aufzählung sollte ein heute modern gewordenes Anwendungsgebiet und „Wettbewerbskriterium" nicht vergessen werden – das Desktop Publishing oder kurz DTP –, das zweifellos dank dem Macintosh zu seiner jetzigen Bedeutung gelangte. Einerseits ist das Mac-typische problemlose Erlernen der Gestaltung von einfacheren Publikationen (Broschüren, Datenblättern) durch Laien auf dem Gebiet des Druckwesens, ja selbst das Einbinden von gescannten hochaufgelösten Bildern in ein Layout und die Ausgabe auf einem Laserdrucker oder gar einer Lichtsatzmaschine relativ einfach möglich. Andererseits kann im professionellen Anwendungsbereich für Verlage und Druckereien nur ein Satzfachmann wirklich zufriedenstellende Produkte mit DTP herstellen. Oft wird in der Literatur (z. B. in /5/) angegeben, daß die Forderungen auch der Profiebene nur durch Macintosh-Systeme in Verbindung mit DTP-Softwarepaketen erfüllt werden (beispielsweise mit dem marktführenden Programm PageMaker von Aldus und Grafikprogrammen wie Adobe Illustrator sowie echten Bildverarbeitungsprogrammen wie dem in Bild 6 gezeigten Image-Studio sowie Freehand). Die drei erfolgreichen **A**-Firmen Apple, Aldus und Adobe waren auf dem DTP-Markt dermaßen überzeugend, daß sich IBM 1987 der von diesen verwendeten (und von Adobe entwickelten) Seitenbeschreibungssprache PostScript anschloß und diese damit zum Quasi-Industriestandard wurde. PostScript erlaubt mit entsprechend ausgerüsteten Laserdruckern professionelle Qualitätsbelichtungen mit bis zu 500 bzw. 1 000 dpi (Punkte pro Zoll). Im Gegensatz zu IBM distanziert sich übrigens die Firma Rank Xerox mit ihren ansonsten ausgezeichneten Laserdruckern von PostScript, ebenso wie ihr DTP-Produkt Ventura Publisher eher Satzfachleute ansprechen will und nicht der Linie des Anfertigens von reprofähigen Druckvorlagen durch „jedermann" folgt. Für DTP auf Mac muß noch die erfolgreiche Software „ReadySetGo!" Version 4.0 von Letraset als Trendsetter erwähnt werden, die den bisherigen PageMaker ablösen könnte (siehe Bild 7).

Die vielfachen weiteren grafischen und insbesondere auch akustischen Fähigkeiten des Macintoshs mit seinen sechs Soundgeneratoren (Verwendung bei Kompositionen, als Sequenzer für Musiker, digitale Tonspeicherung und -auswertung in Stereo mit CD-Qualität bei 44,1 kHz Abtastrate, Tonsynthesen für alle möglichen Instrumente usw.), die Sprachsynthese (Vorlesen von Text, auch direkt beim Eintippen), die OCR-Fähigkeit, das heißt Zeichenerkennung/-lesen von bildhaften (gescannten) Textvorlagen usw. können hier nicht behandelt werden. Auf all diesen Gebieten gab es und gibt es Vorbildwirkungen für Speziallösungen oder ganze Systeme. Selbst die sogenannte Teachware in Form einer trickfilmartig aufgebauten Schulungsdiskette zum Erlernen der Macintoshbedienung in 2 bis 3 Stunden ist gewissermaßen ein Maßstab für die Gestaltung guter Schulungsunterlagen.

An dieser Stelle nun müßte ein Ausblick kommen, wie es weitergehen könnte. Mit Bezug auf den von Steven Jobs' neuer Firma NeXT produzierten Computer Cube ließe sich fragen: „What's NeXT?". Nun, NeXT ist *eine* der möglichen Antworten, um mit eben diesem neuen „schwarzen Computerwürfel" eine fortschrittliche Entwicklungsrichtung einzuschlagen. Aber der kommerzielle Erfolg bzw. eine breitere Durchsetzung der neuen Supermaschine wird sicher noch Jahre dauern. Jedoch wird eine andere Entwicklung, die im Herbst 1987 von Bill Atkinson geschaffene Softwareumgebung Hyper Card, voraussichtlich in der nächsten Zeit wieder Maßstäbe setzen. Es ist ein neues Konzept, das nicht einfach nur Anwendersoftware oder „Büro-Expertensystem" genannt werden kann; und es bereitet offensichtlich selbst den Erfindern Mühe, in wenigen Sätzen auszudrücken, was das nun eigentlich ist. Vielleicht könnte man HyperCard als ein in Stapeln organisiertes relationales Datenbanksystem bezeichnen, das alle denkbaren Medien (einschließlich Bild, Animation, CD-ROM, Video-Platte, Ton, Sprache, Telekommunikation usw.) einschließt und zudem mit einer eigenen, sehr leicht erlernbaren Hochsprache durch jeden Benutzer sehr einfach an eigene Bedürfnisse angepaßt werden kann. HyperCard besitzt zudem eine eigene Benutzeroberfläche (als Beispiel siehe Bild 8), und es kann außerdem multitaskingartig mit jeder anderen Anwendersoftware des Mac zusammenarbeiten. Es sind aus der Literatur Bestrebungen bekannt geworden, HyperCard mit einigen Einschränkungen nun auch für die MS-DOS-Welt zu schaffen.

Man kann jedenfalls überzeugt sein, daß auch in der Zukunft noch manche Überraschung aus der Richtung Apple Macintosh und NeXT kommen wird.

Grundmodell	Macintosh Plus	Macintosh SE 2DD	Macintosh SE HD	Macintosh II	Macintosh IIx	Macintosh SE/30	Macintosh IIcx
Bauform	siehe Bild 1	ähnlich Bild 1, aber mit 2 Laufwerkschlitten	ähnlich Bild 1	siehe Bild 2	ähnlich Bild 2	siehe Bild 4	individuell konfigurierbares, modulares Konzept; siehe Bild 3
Prozessor Math. Koprozessor Taktfrequenz	MC 68000 ohne 8 MHz	MC 68000 ohne 8 MHz	MC 68000 ohne 8 MHz	MC 68020 68881 16 MHz	MC 68030 68882 16 MHz	MC 68030 68882 16 MHz	MC 68030 68882 16 MHz
Standard-RAM RAM ausbaubar bis ROM	1 MByte 2,5 oder 4 MByte 128 KByte	1 oder 2 MByte 4 MByte 256 KByte	1 oder 2 MByte 4 MByte 256 KByte	1 MByte 8 MByte auf dem Board 2 GByte in Slots 256 KByte	4 MByte 8 MByte, o. Ang., Erweit. in Slots 256 KByte	2 MByte 8 MByte, 4 GByte in Slots 256 KByte	2 MByte 8 MByte, 256 KByte
Anschlüsse	2xseriell RS 422 1xSCSI:bis 7 Peripheriegeräte 1 × Mausschnittstelle 1xAudio: Mono ohne Erweiterung intern	2xseriell RS 422 1xSCSI:bis 7 Peripheriegeräte für Maus, Tastatur: 1 × ADB:max. 16 Geräte 1xAudio:Mono 1 × interne Erweiterung 96 pins	2xseriell RS 422 1xSCSI:bis 7 Peripheriegeräte für Maus, Tastatur: 1 × ADB:max. 16 Geräte 1xAudio:Mono 1 × interne Erweiterung 96 pins	2xseriell RS 422 1xSCSI:bis 7 Peripheriegeräte für Maus, Tastatur: 2 × ADB:max. 16 Geräte 1xAudio:Stereo 6xNuBus-Erweiterung	2xseriell RS 422 1xSCSI:bis 7 Peripheriegeräte für Maus, Tastatur: 2 × ADB:max 16 Geräte 1xMini-jack-Stereoton 6xNuBus-Erweiterung	ADB 4-Kanal-8bit-Stereo NuBus:030 Direct Slot	ADB 4-Kanal-Stereoton 3xNuBus-Erweiterung
Floppylaufwerk 3,5 Zoll	1 × 800 KByte	2 × 800 KByte	1 × 800 KByte	1 oder 2 × 800 KByte	1 oder 2 × 1,44 MByte FDHD	1 × 1,44 MByte FDHD	1 × 1,44 MByte FDHD
integr. Festplatte	ohne	ohne	20 oder 40 MByte	40 oder 80 MByte	40 oder 80 MByte	40 MByte	40 oder 80 MByte
Bildschirm	integr., 9", s/w, 512×342 Pixel, hochauflösend	integr., 9", s/w, 512×342 Pixel, hochauflösend	integr., 9", s/w, 512×342 Pixel, hochauflösend	12", s/w, 640×480 oder 13", color, 640×480 Pixel	12", s/w, 640×480 oder 13", color, 640×480 Pixel es gibt diverse Bildschirme von Fremdherstellern, bis 21", 1152 × 870 Pixel	integr., s/w, 512×342 Pixel, hochauflösend	12", s/w, 640×480 oder 13", color 640×480 Pixel
Bemerkungen	historisch ältestes, preisgünstigstes und wahrscheinlich verbreitestes Grundgerät	mit den beiden SE-Grundtypen kann Anpassung an viele Anwendungsfälle erfolgen. Aufbau der SE modularer, integrierter als Plus bei Erhaltung von dessen Kompaktheit und Vorteilen, auch Transport leicht möglich		Grundmodell für Anwendung in Industrie	leistungsfähigstes Modell, nutzt jedoch noch nicht alle Vorteile von 68030 und Koprozessor voll aus, so daß Steigerungen z. T. unter theoretischen Erwartungen liegen zugunsten einer Softwarekompatibilität innerhalb der Mac-Familie	gilt als neue Kompaktform des Mac IIx im bewährten SE-Format	paßt zwischen den langsameren Mac II und den „großen" Mac IIx, dabei völlig neue Bauform: schmaler und paßt direkt unter Apple-Bildschirm, hochkantiger Betrieb ist möglich (Tower)

Literatur

/1/ Duff, C. B.: Introducing the Macintosh. McGraw-Hill Book Company, New York 1984

/2/ Herge, H.: Warum „1984" nicht stattfand. MACup, Hamburg 5 (1989) 1, S. 20

/3/ Meyers, P.: The AT Animates the Airwaves. PC World 4 (1986) 2, S. 146

/4/ Pearlman; Dara u. a.: Operating Systems. PC Magazine 5 (1986) 4, S. 211

/5/ Pfirstinger, R.: DTP auf Mac und IBM-PC. CHIP 10 (1988) 2, S. 208

/6/ Gergely, M.: 13 Trends in der Computerwelt. tech-trend, Wien 12/1988

Arithmetik für die Meßtechnik

Heiko Wand
VEB Mikroelektronik „Karl Marx" Erfurt

In /1/ wurde eine Arithmetikeinheit auf der Basis des 16-Bit-Mikroprozessors U 8000 vorgestellt, welche K 1520-buskompatibel ist und folgende Voraussetzungen erfüllt:

1. Grundrechenarten in verschiedenen Zahlenbereichen
2. trigonometrische Funktionen
3. Sonderfunktionen (Negation, Schiebefunktion, ...)
4. Konvertierung von Zahlensystem in Zahlensystem, Konvertierung von Zahlensystem in Zeichenfolge (String)
5. Kettenverarbeitung von Funktionen.

Bei Einsatz einer solchen Arithmetik (ARI) in der Meßtechnik sind zusätzlich noch folgende Forderungen zu stellen:

6. Maßeinheitenbehandlung
7. Verarbeitung von logischen Größen
8. komfortable Ein-/Ausgabekonvertierung.

Damit ergab sich die Notwendigkeit, die in /2/ beschriebene Arithmetik zu erweitern.

Zur Erfüllung des Punktes 6 muß der Formataufbau geändert werden. Im *Gleitkommamodul* von /2/ wurde die Gleitkommazahl (Real-Zahl) f wie folgt zerlegt:

```
           1. Wort                2. Wort
f =   eeeeeeeeeeeeeeee      mmmmmmmmmm
           3. Wort
        mmmmmmmmmm
```

Durch Einschränkung des Exponenten auf 12 Bit ergab sich die Möglichkeit, 4 Bit für die Kodierung von Maßeinheiten im gegebenen Format zu nutzen.

Damit ergibt sich folgendes neue Format:

```
              1. Wort                  2. Wort
f =   cccceeee eeeeeeee      mmmmmmmm
              3. Wort
          mmmmmmmm
```

(c = Maßeinheit, e = Exponent, m = Mantisse)

Die daraus folgende Beschränkung des Wertebereiches ist nach Berechnung des Wertevorrates $f = m \times 2^{exp.}$ mit

$0{,}5 \leqq m \leqq 1{,}0 - (2^{-31})$ oder $m = 0$ und

$-2\,048 \leqq exp. \leqq 2\,047$

$\rightarrow 0{,}154717307 \cdot 10^{-616} < |f|$

$|f| < 1{,}615850261 \cdot 10^{616}$

für den Einsatz in der Meßtechnik unerheblich. (Exponent sowie Mantisse werden im Zweierkomplement dargestellt.)

Folgende Maßeinheiten sind möglich: Ampere, Volt, Farad, Watt, Sekunde, Hertz, Ohm und ohne Maßeinheit.

Logische Größen werden intern als 32-Bit-Wert dargestellt. Bei der Anwendung arithmetischer Operationen oder Funktionen auf Logical-Werte werden sie vorzeichenbehaftet betrachtet.

Für die Spezifizierung des *Konvertierungsablaufes* wurde das Register R15 /2/ zusätz-

lich als Formatregister genutzt. Folgende Steuerinformationen sind im Formatregister enthalten:

- Formatangabe (binär, oktal, hexadezimal, dezimal)
- Multiplikationsfaktoren (pico ... Giga, wahlfrei)
- Gesamtzeichenzahl
- Stellen nach den Komma
- Zieltyp (string, integer, logical, real)
- Zeichenzeiger (Pointer) nach Konvertierung.

Die Eingabekonvertierung ohne Format und ohne Zieltyp wandelt einen String in eine der Zahlentypen *integer, logical* oder *real.* Die Zeichenfolge wird dabei auf konvertierbare Kombinationen untersucht. Als Resultat der Konvertierung wird im Formatregister der entstandene Zahlentyp sowie der Pointer eingetragen. Der Pointer zeigt nach der Konvertierung auf das erste nicht konvertierte Zeichen.

Die Eingabekonvertierung ohne Format und mit Zieltyp wird intern durch das Ausführen einer Eingabekonvertierung ohne Format und ohne Zieltyp mit anschließender Typenwandlung in den geforderten Zieltyp realisiert.

Bei der Eingabekonvertierung mit Format und mit Zieltyp ist das gewünschte Format zusammen mit der Gesamtzeichenzahl vor der Befehlsausführung im Formatregister einzutragen. Obwohl der Typ des Ergebnisses festliegt, wird er wie bei der Eingabekonvertierung ohne Zieltyp als Ergebnis im Formatregister eingetragen. Im Gegensatz zu den nicht formatierten Eingabekonvertierungen wird die im Formatregister angegebene Gesamtzeichenzahl in die Konvertierung einbezogen. Der Pointer zeigt nach der Konvertierung auf das erste nicht konvertierte Zeichen.

Ausgabekonvertierung ohne Format (Standardformat):

Integer-Zahlen werden insgesamt mit 8 Zeichen ausgegeben. Die letzten zwei Zeichen sind immer Leerzeichen, um die Kompatibilität zur Ausgabe von Real-Zahlen mit Maßeinheiten und Multiplikationsfaktor zu gewährleisten.

Logical-Zahlen werden immer durch eine Folge von 32 Binärziffern (0/1) sowie durch die Kennung B dargestellt.

Real-Zahlen werden in der Regel im Festkommaformat, das heißt ohne Exponent mit insgesamt acht Zeichen ausgegeben. Die letzten zwei Zeichen beinhalten immer Multiplikationsfaktor und Maßeinheit. Kann der Wert der Real-Zahl nicht in diesem Format dargestellt werden, so wird die Real-Zahl im Gleitkommaformat mit Exponent mit insgesamt 15 Zeichen einschließlich Multiplikationsfaktor und Maßeinheit ausgegeben.

Für die Ausgabekonvertierung mit Format ist im Formatregister eine entsprechende Information (Format, Gesamtzeichenzahl, Stellen nach dem Komma, Multiplikationsfaktor usw.) anzugeben.

Als Ergebnis der Ausgabekonvertierung wird der Typ *string* eingetragen. Die Quellüberarbeitung erfolgte auf der Basis von Software, die unter SCPX lauffähig ist.

Literatur

/1/ Fey, P.; Kriesten, S.; Rieken, R.: Frei programmierbarer Arithmetikmodul für K 1520. Radio, Ferns., Elektron., Berlin 33 (1984) 8, S. 483

/2/ Anwenderbeschreibung zum Arithmetikprozessor. Technische Universität Karl-Marx-Stadt 4/87

✉ KONTAKT ☎

VEB Mikroelektronik „Karl Marx" Erfurt, Bereich Testerentwicklung, Rudolfstraße 47, Erfurt, 5020; Tel. 58 24 39

Wegbereiter der Informatik



LUDWIG EDUARD BOLTZMANN

∗1844 Wien
† 1906 Duino bei Triest

Der österreichische Physiker und Mathematiker L. E. Boltzmann hat unter anderem bei Joseph Loschmidt (1821–95) und Josef Stefan (1835–93) studiert, 1866 an der Universität Wien promoviert und danach dort eine Assistentenstelle bekleidet. Im Jahre 1869 wurde er Physikprofessor an der Universität Graz und 1873 Mathematikprofessor in Wien, kehrte aber 1876 wieder nach Graz zurück. Weitere Stationen waren die Universitäten in München (ab 1889), Wien (1894), Leipzig (1900) und wieder Wien (1902), wo er jeweils als Professor für theoretische Physik wirkte und auch Naturphilosophie lehrte.

Boltzmanns Lebensarbeit galt (gemäß einer Einschätzung des Physikers Arnold Sommerfeld) der Einordnung der Thermodynamik in das Weltbild der klassischen Mechanik. Er schuf die Grundlagen zu einer umfassenden Statistik des physikalischen Geschehens, wobei er den sonst üblichen Wahrscheinlichkeitsbegriff schärfer faßte. Das von ihm aufgestellte *Boltzmannsche H-Theorem* zeigt auf, daß das Weltgeschehen von unwahrscheinlichen Anfangszuständen zu wahrscheinlichen Endzuständen fortschreitet, wodurch der einseitig gerichtete Verlauf thermodynamischer Prozesse seine Erklärung findet.

Durch seine 1873 erfolgte Bestimmung der Dielektrizitätskonstanten von Gasen lieferte er eine erste experimentelle Bestätigung für eine der Voraussagen der Maxwellschen Lichttheorie. Ihm gelang es, unter Heranziehung statistischer Rechenverfahren einen grundlegenden Zusammenhang zwischen der thermodynamischen Entropie S und der Wahrscheinlichkeit W der jeweiligen molekularen Bewegungszustände in einem gasförmigen Stoffsystem zu finden $S = k \cdot \ln W$ (dabei ist die Naturkonstante k die sogenannte Boltzmann-Konstante). Diese Formel ist übrigens auch auf seinem Ehrengrab auf dem Zentralfriedhof in Wien festgeschrieben. Boltzmann war Mitglied der Akademien Amsterdam, Berlin, Göttingen, London, New York, Paris, Petersburg, Rom, St. Louis, Stockholm, Turin, Upsala, Washington und Ehrendoktor der Universität Oxford.

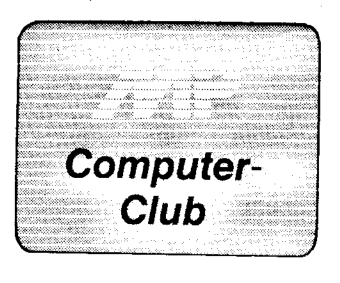
Daß L. E. Boltzmann im Zusammenhang mit der modernen Computertechnik zu nennen ist, hat sich erst in neuester Zeit ergeben, und zwar seit es den *Boltzmann-Rechner* gibt, der 1983 von Hinton und Rajnowski erfunden wurde und indessen zu einem feststehenden Begriff geworden ist. Die statistischen Methoden von Boltzmann haben nämlich dazu Anlaß gegeben, ein neues Rechnerprinzip zu entwickeln, das von der bekannten Struktur des von-Neumann-Rechners völlig verschieden ist. Der Boltzmann-Rechner (ein Vertreter der Knotenrechner oder Neuronrechner) ist ein „lernfähiges", parallel verbundenes, stochastisches Netzwerk mit folgenden Merkmalen:

- Das Netzwerk besteht aus adaptiven Elementen (Knoten) in hierarchischer Organisation und enthält 3 Funktionsebenen: Eingangsebene, Ausgangsebene und dazwischen eine interne Ebene. Zwischen den Knoten in diesen Ebenen existieren bidirektionale Verkopplungen, deren Bewertung von stochastischen Einflüssen abhängt.
- Das System läuft abwechselnd in einer Lernphase (mit festen Input-Output-Relationen), einer Testphase (mit festem Input und freiem Output) und einer Korrekturphase, in der die Kopplungsmatrix erforderlichenfalls korrigiert wird.
- Das System wird solange „belehrt" (ein Durchlauf der drei Phasen ist ein Belehrungszyklus), bis es bei gegebenem Input den gewünschten Output erzielt.

Ein solcher Belehrungsprozeß fußt auf Analogiebetrachtungen zur statistischen Thermodynamik, indem ein bestimmtes Minimalkriterium herangezogen wird, und zwar ist die Zielgröße der Belehrung beim Boltzmann-Rechner das „Energieminimum".

Aus alledem geht bereits hervor, daß der Boltzmann-Rechner kein Universalrechner sein kann, sondern einen Spezialrechner für bestimmte Aufgabenklassen – zum Beispiel für Mustererkennung – darstellt.

Dr. Klaus Biener



Bildschirmsteuerung des PC 1715

Die hardwareabhängigen Steuerzeichen für Bildschirmsonderzeichen liegen zwischen 128 und 255. Man kann sich durch direktes Einschreiben in den Bildwiederholspeicher experimentell am konkreten Bildschirm die Wirkung ansehen. Das kleine Basic-Programm in Bild 1 kann dazu als Grundlage dienen.

```
1   REM BS-Steuerz.  zwischen 128 und 255:
2   PRINT CHR$(12)
10 K%=4
20 S%=&HF800+(K%+1)*80-8
30 FOR I%=128 TO 255
40 S%=S%+8
50 POKE S%+7,128: POKE S%,I%
60 C%=(I%+2) MOD 10
70 IF C%=0 THEN K%=K%+1
80 PRINT CHR$(27)+CHR$(128+K%)+CHR$(129+
C%*8);I%
90 PRINT CHR$(27)+CHR$(128+0)+CHR$(128+0);
"Cursor"
95 NEXT I%: END
```

Bild 1 Basic-Programm

Von diesen Steuerzeichen seien folgende sieben Funktionen als Attribute ausgewählt:

Normale Helligkeit	**80H**
Intensive Helligkeit	**81H**
Hell blinkend	**83H**
Invers	**90H**
Invers hell	**91H**
Invers blinkend	**92H**
Invers blinkend hell	**93H**

Um nun eine Unabhängigkeit vom speziellen CP/M-Betriebssystem zu erreichen, wird durch Vereinbarung von sieben Attributen gemäß obiger Aufzählung in Form von sieben beliebig wählbaren ASCII-Zeichen, die während ihrer Funktion jedoch nicht gleichzeitig in Bildschirmtexten vorkommen dürfen, erst einmal der Bildaufbau allein mittels gültiger ASCII-Zeichen vorgenommen. Danach wird ein Unterprogramm aufgerufen, das den Bildwiederholspeicher von hinten nach vorn durchsucht und dabei diese Attribut-Kennzeichen in echte Steuerzeichen (80H ... 93H) umwandelt, wodurch die beabsichtigten Bildschirmattribute aktiviert werden. Vorteile dieser Methode sind neben der Betriebssystem-Unabhängigkeit die variable Gestaltung des Bildaufbaus auch von „text ... endtext"-Abschnitten und vor allem das Wegfallen von lästigem Aufblitzen bei Invers-Umschaltung, da von hinten her eingeschaltet wird.

In Bild 2 und Bild 3 sind die Unterprogramme für dBase II und für Turbo-Pascal gezeigt.

Für BAB1 muß in der vorletzten Zeile (Zeile 23 bzw. Zeile 12) **128, 7, 127, 255** gegen **0, 4, 255, 255** ausgetauscht werden.

Thomas Bauer

```
 1: * XVDO.CMD vom: 02.03.88
 2: * Anwendung:
 3: * =========
 4: * Es werden 7 Bildschirm-"Farben" vereinbart fuer:
 5: * STORE '0' TO NORM
 6: * STORE '1' TO HELL
 7: * STORE '2' TO HBLINK
 8: * STORE '3' TO INVERS
 9: * STORE '4' TO INVHEL
10: * STORE '5' TO INVBLI
11: * STORE '6' TO HBLINV
12: * STORE NORM+HELL+HBLINK+INVERS+INVHEL+INVBLI+HBLINV TO MStz
13: * ERASE
14: * @ 10,0 say HBLINK+"Hell Blinkend ===>"+NORM
15: * DO XVDO
16: * CALL MStz
17: * @ 13,10 say INVERS+"Inverse Darstellung . . .  "+NORM
18: * CALL MStz
19: POKE   181,126,254,7,192,35,1,7,0,17,234,0,237,176;
20: ,42,250,0,237,91,248,0,126,229,205,217,0,225,32,1;
21: ,119,43,27,122,179,32,241,201,33,234,0,1,7,0,237;
22: ,177,192,43,229,221,225,221,126,7,201,0,0,0,0,0
23: POKE 239,0,0,128,129,131,144,145,146,147,128,7,127,255;
24: ,0,4,255,255
25: SET CALL TO   181
26: RETURN
27: * XVDO.CMD  ab Adresse: 00B5 bis: 00FF
```

Bild 2 dBase II-Unterprogramm zur Aktivierung der vereinbarten Attribute auf dem Bildschirm des PC 1715

```
 1:  0 PROCEDURE XVIDEO(VAR STZ:BStz);
 2:  0 (* TYPE BStz=ARRAY[1..7] OF CHAR;
 3:  0     VAR  STZ : BStz;
 4:  0 Die 7 vereinbarten Steuerzeichen werden in der Reihenfolge
 5:  0 STZ[1] .. STZ[7]  --> 80h, 81h, 83h, 90h, 91h, 92h, 93h
 6:  0 umgewandelt.                                             *)
 7:  0 BEGIN {Modul XVIDEO} INLINE(
 8:  1 $2A/STZ/$01/$07/$00/$11/*+44/$ED/$B0/$2A/*+56/$ED/$5B/
 9:  1 *+50/$7E/$E5/$CD/*+13/$E1/$20/$01/$77/$2B/$1B/$7A/$B3/
10:  1 $20/$F1/$C9/$21/*+16/$01/$07/$00/$ED/$B1/$C0/$2B/$E5/
11:  1 $DD/$E1/$DD/$7E/$07/$C9/$60/$22/$5E/$7E/$23/$7B/$27/
12:  1 $80/$81/$83/$90/$91/$92/$93/$88/$80/$07/$7F/$FF/$88/
13:  1 $00/$04/$FF/$FF)
14:  1 END;  {Modul XVIDEO}
```

Bild 3 INLINE-Prozedur für Turbo-Pascal zum Aktivieren von 7 nutzereigenen Bildschirmattributen am PC 1715

Umcodieren der Steuertasten unter MS-DOS

MS-DOS verfügt über eine große Anzahl von Funktions- und Funktionssteuertasten. Dabei liefert die Tastatur für Funktionstasten als erstes den Code 0 (null), als zweites den Scancode der Taste. Gegenüber Programmen unter CP/M folgt daraus die Umstellung, daß für bestimmte Funktionen, die häufig auch in Anwenderprogrammen benutzt werden, nicht mehr der ASCII-Code der Ctrl-Tastenbetätigung zu benutzen ist. Beide Tastenbetätigungen zuzulassen, also sowohl die herkömmlichen Ctrl-Tasten als auch die DOS-Funktionssteuertasten, erfordert zusätzlichen Aufwand. Um Änderungen in vorhandenen Programmen so gering wie möglich zu halten, wurde für Turbo-Pascal Version 4.0 folgende Routine geschrieben, die bei Auftreten des Tastaturcodes 0 den nachfolgenden Scancode in die entsprechende Ctrl-Tastenbelegung umwandelt.

```
{ Prozedur FKTASTE zum Umkodieren von
Funktionstasten }
procedure fktaste (VAR ta:char);
begin
ta:=readkeyi       {Erneuerung Aufruf
                   Tastatur}
case ta of         {Umkodieren}
  #72:ta:=^E;      {Kursor hoch}
  #80:ta:=^X;      {Kursor tief}
  #75:ta:=^S;      {Kursor links}
  #77:ta:=^D;      {Kursor rechts}
  #73:ta:=^R;      {Seite zurück, PgUp}
  #81:ta:=^C;      {Seite vorwärts, PgDn}
  #82:ta:=^V;      {Einfügen, INS}
  #83:ta:=^G;      {Löschen, DEL}
end;
end;               {Ende der Prozedur}
{Ausschnitt aus einem aufrufenden Programm}
...
key:=readkey;   {Tastencode holen}
if key=0 then fktaste(key);  {Ergänzender Aufruf FKTASTE}
case key of
 ^X:;
 ^E:;
 ^S:;
 ^D:;
end;
...
```

Michael Lennartz

TERMINE

Symposium „Industriebussysteme"

WER? Bezirksvorstand der Kammer der Technik Karl-Marx-Stadt, Technische Universität Karl-Marx-Stadt und WGMA der KDT, FA 2 und FA 6
WANN? 29. November 1989
WO? Karl-Marx-Stadt
WAS?
- Stand der Entwicklungsarbeiten in der DDR auf dem Gebiet der prozeßnahen Bussysteme, insbesondere PROFI-Bus
- Überblick zu Industriesystemen
- Hard- und Softwareentwicklungen zur Implementierung der ISO-Schichten 1, 2 und 7

WIE? Rückfragen oder Teilnahmemeldungen richten Sie bitte an: Kammer der Technik, BVo Karl-Marx-Stadt, Tagungsorganisation, PSF 504, Karl-Marx-Stadt, 9010; Tel. 62141

Dr. Buschbeck

1. DDR-Symposium zur Programmiersprache Ada

WER? VEB Leitzentrum für Anwendungsforschung Berlin
WANN? 29. November 1989, 10.30 Uhr
WO? Berlin, VEB LfA (Haus B, Raum 003)
WAS?
- Internationaler Entwicklungsstand
- Ada-Entwicklungsarbeiten in der DDR
- Programmieren in Ada (Leistungen und Konzepte der Sprache)
- Ada-Anwendungen

WIE? Für eigene Beiträge bitte Vortragskurzfassung mit Angabe der Zeitdauer zuschicken. Teilnahmemeldungen richten Sie bitte an: VEB Leitzentrum für Anwendungsforschung – Softwarebetrieb CAD/CAM – Abt. F13, Kollege Grützbach, Jacques-Duclos-Straße 47/52, Berlin, 1156; Tel. 378030

Dr. Warnow

Kleines Lexikon der Mikrorechentechnik

N
wie Nadeldrucker



Zeichnung: Dahmen

Komfortable formatierte Eingabe in Basic

Bedienerfreundliche Programme sind dialogorientiert. Nutzereingaben sollten so früh wie möglich überprüft werden, um Programmabstürze zu vermeiden. Eine freizügige Kursorsteuerung von Eingabefeld zu Eingabefeld zur leichten Korrektur falscher Eingaben ist dabei sehr wichtig.

Bei SUPI.BAS handelt es sich um ein universell verwendbares Teilprogramm zur formatierten Eingabe, das aus etwa 80 Zeilen Basic-Qelltext besteht und unter Beachtung der Übergabe und Rückkehrbedingungen in Basicprogramme eingebunden werden kann. Dabei entsteht der größte Vorteil für den Bediener natürlich dann, wenn in einem Programm (oder Unterprogramm) eine größere Zahl von verschiedenartigen Eingaben notwendig ist, die dann gleichzeitig auf einem (nicht unzeitgemäß rollenden) Bildschirm dargestellt und abgefordert werden kann.

Dieses Programm benötigt in compilierten Programmen etwa 3,2 KByte. Seit 1986 läuft es in mehreren Anwenderprogrammen um eine Datumsroutine ergänzt ohne Beanstandung.

Felder werden durch Doppelpunkte begrenzt, in numerischen Feldern wird der Dezimalpunkt auch ohne Variableninhalt dargestellt.

Für die richtige Wahl der Feldlänge SL und damit der Maskengröße (größer oder gleich der Variablenlänge im Hauptprogramm) ist ebenfalls der Programmierer verantwortlich.

Bei numerischen Eingaben ist in der ersten Position nur '–' oder Leerzeichen zugelassen. Bei Übergabe der Feldlänge ist die Vorzeichenstelle (und der Dezimalpunkt) zu berücksichtigen. Die erste Eingabe eines Dezimalpunktes führt zur *Zentrierung* der Eingabe an der bei der Übergabe mit der Variablen 'SD' vorgesehenen Stelle im Eingabefeld.

Sollten numerische Variablen doppelter Genauigkeit nötig werden, müssen die Variablen SP, SRT im numerischen Teil des Moduls durch SP#, SRT # ersetzt werden.

Bild 1 enthält die universell einsetzbare Eingaberoutine (SUPI).

Bild 2 zeigt ein Testprogramm, das zusammen mit dem Programm nach Bild 1 eine sinnvolle Einbindung von SUPI in Anwenderprogramme demonstriert.

Vor Aufruf des Moduls mit GOSUP <Zeilennummer> müssen folgende Variablen spezifiziert werden:

SZ%, ST% Zeile, Spalte der ersten Feldposition
SL Feldlänge gesamt (BEACHTE Variablenlänge!)
SD Anzahl der Dezimalstellen

```
7490 REM --- FORMATIERTE EINGABE (SUPI vom 31.08.88) ---
7495     - VORBEREITUNG
7500 SRT$="":SR$="":SDEZ$="":SV=0          'INITIALISIERUNG
7510 GOSUB 8500                             'CURSORPOSITIONIEREN
7520 ON STP GOSUB 7550,7860
7530 RETURN 'AUSTRITT AUS SUPI
7540 'VORBEREITUNG NUMERISCH
7550 IF SD=0 THEN PRINT":"+SPACE$(SL)+":" ELSE PRINT":"
     +SPACE$(SL-SD-1)+"."+SPACE$(SD)+":"
7560 ST%=ST%+1:GOSUB 8500
7570 'BEREITS EINGEGEBENER WERT (NUMERISCH)
7580 IF SD=0 THEN SU$=STRING$(SL,"#") ELSE
     SU$=STRING$(SL-SD-1,"#")+"."+STRING$(SD,"#")    'USING-MASKE
7590 IF SD=0 THEN SRT$=SPACE$(SL) ELSE SRT$=SPACE$(SL-SD-1)+"."
     +SPACE$(SD)
7600 IF SP<>0 THEN IF SD<>0 THEN
     SRT$=LEFT$(SPACE$(SL-SD-1-LEN(STR$(INT(SP))))+
     STR$(SP)+STRING$(SD+1,"0"),SL):MID$(SRT$,SL-SD,1)=".":
     ELSE SRT$=SPACE$(SL-LEN(STR$(SP)))+STR$(SP)
7610 IF SP<>0 AND ABS(SP)<1 THEN SRT$=SPACE$(SL-LEN(STR$(SP)))
     +STR$(SP)
7620 IF SP<>0 THEN PRINT USING SU$;SP
7630   - NUMERISCHE EINGABE
7640 GOSUB 8500                                  'CURSORPOSITIONIEREN
7650 SI=1
7660 WHILE SI<>SL+1
7670  GOSUB 8020                           'EINGABE/CURSORSTEUERUNG
7680 IF SV>0 THEN 7790                       'WEITERE VERARBEITUNG
7690 IF SI=1 AND SR$<>"-" AND SR$<>" " THEN SI=2:
     MID$(SRT$,1,1)=" "
7700 IF SR$<>"-" THEN 7720                  '1. ZEICHEN VORZEICHEN
7710 IF SI=1 THEN 7770 ELSE GOTO 7670
7720 IF SR$<>"." THEN 7750                      'KOMMA EINGEGEBEN?
7730 IF SDEZ$="E" THEN 7670           'KOMMA BEREITS EINGEGEBEN?
7740 SDEZ$="E":SRT$=SPACE$(SL-SD-SI)+SRT$:
     SRT$=LEFT$(SRT$,SL-SD)+RIGHT$(SRT$,SD):SI=SL-SD:GOTO 7770
7750 IF (SR$<"0" OR SR$>"9") AND SR$<>" " THEN 7670
7760 IF SI=SL-SD AND SR$<>"." THEN IF SD<>0 THEN SI=SI+1:
     SDEZ$="E"
7770 MID$(SRT$,SI,LEN(SR$))=SR$
7780 GOSUB 8500:PRINT SRT$:SK%=ST%+SI:GOSUB 8510
7790 SI=SI+1
7800 WEND
7810 IF SV=0 THEN SV=1
7820 IF LEN(SRT$)>0 THEN SRT=VAL(SRT$) ELSE SRT=0
7830 IF SE=1 THEN IF LEN(SRT$)=0 OR SRT$=SPACE$(SL-SD-1)+"."
     +SPACE$(SD) OR SRT$=SPACE$(SL) THEN 7640
7840 RETURN   'ENDE NUMERISCH
7850 'VORBEREITUNG CHARAKTER
7860 PRINT":"+SPACE$(SL)+":"
7870 ST%=ST%+1:GOSUB 8500
7880 'BEREITS EINGEGEBENER WERT(CHARAKTER)
7890 IF SP$<>"" THEN SRT$=SP$:PRINT SP$                 'CHARAKTER
7900 '   - CHARACTER EINGABE
7910 GOSUB 8500                            'CURSORPOSITIONIERUNG
7920 FOR SI=1 TO SL                              'EINGABESCHLEIFE
7930 GOSUB 8020                         'EINGABE/CURSORSTEUERUNG
7940 IF SV>0 THEN 7970
7945 IF SG%=1 AND SR$>CHR$(96) THEN SR$=CHR$(ASC(SR$)-32)
7950 IF SI<=LEN(SRT$) THEN MID$(SRT$,SI,1)=SR$
     ELSE SRT$=SRT$+SR$
7960 GOSUB 8500:PRINT SRT$:SK%=ST%+SI:GOSUB 8510
7970 NEXT SI
7980 IF SV=0 THEN SV=1
7990 IF SE=1 THEN IF LEN(SRT$)=0 OR SRT$=SPACE$(SL) OR SRT$=" "
     THEN 7920
8000 RETURN   ENDE CHARAKTER
8010 '  EINGABESTEUERUNG UND CURSORPOSITIONIERUNG
8020 SV=-1
8030 WHILE SV=-1
8040 FOR SJ%=0 TO 1:SR$=INKEY$:SJ%=-1*(SR$<>""):NEXT
8050 IF SR$=CHR$(13) OR SR$=CHR$(6) THEN SI=SL:
     SV=1                                          'RETURN,^F
8060 IF SR$=CHR$(27) THEN SI=SL:SV=3                'ABBRUCH
8070 IF SR$=CHR$(8) OR SR$=CHR$(127) THEN SI=SI-1:
     ON STP GOSUB 8210,8200
8080 IF ASC(SR$)>=32 AND ASC(SR$)<127 THEN
     SV=0                                 'KEIN STEUERZEICHEN
8090 IF SR$=CHR$(19) THEN SI=SI-1:GOSUB 8170              '^S
8100 IF ASC(SR$)=4 THEN SI=SI+1:GOSUB 8180                '^D
8110 IF ASC(SR$)=20 THEN SI=1:
     ON STP GOSUB 8240,8270:GOSUB 8500                    '^T
8120 IF ASC(SR$)=1 THEN SI=SL:SV=2                        '^A
8130 IF ASC(SR$)=5 THEN SI=SL:SV=4                        '^E
8140 IF ASC(SR$)=24 THEN SI=SL:SV=5                       '^X
8150 WEND
8160 RETURN  RUECKKEHR AUS EINGABESTEUERUNG
8170 SK%=ST%-1+SI:GOSUB 8510:IF SI<1 THEN SI=SL:SV=2:RETURN
8180 SK%=ST%-1+SI:GOSUB 8510:IF SI>SL THEN SI=SL:SV=1:RETURN
8190 ' BACKSPACE UND DELETE
8200 IF SI<1 THEN SI=SL:SV=2 ELSE SRT$=LEFT$(SRT$,SI-1):
     GOSUB 8220:RETURN
8210 IF SI<1 THEN SI=SL:SV=2 ELSE MID$(SRT$,SI,1)=" ":
     GOSUB 8220:RETURN
8220 SK%=ST%-1+SI:GOSUB 8510:IF SI=SL-SD AND STP=1 THEN
     SDEZ$="":MID$(SRT$,SI,1)=".":PRINT".";SR$;
     ELSE PRINT" ";SR$;:RETURN
8230 ' ^T FELD LOESCHEN
8240 GOSUB 8500:IF SD=0 THEN PRINT SPACE$(SL)
     ELSE PRINT SPACE$(SL-SD-1)+"."+SPACE$(SD)
8250 IF SD=0 THEN SRT$=SPACE$(SL) ELSE SRT$=SPACE$(SL-SD-1)+"."
     +SPACE$(SD)
8260 SDEZ$="":RETURN
8270 GOSUB 8500:PRINT SPACE$(SL):SRT$="":RETURN
8490 REM ---KURSORPOSITIONIERUNG--
8500 PRINT CHR$(27)+CHR$(128+SZ%)+CHR$(128+ST%);:RETURN
8510 PRINT CHR$(27)+CHR$(128+SZ%)+CHR$(128+SK%);:RETURN
```

***Bild 1** Universelle Eingaberoutine SUPI*

```
10 REM TESTPROGRAMM
20 WIDTH 255  wichtig fuer korrekte Kursorpositionierung
30 REM LEERMASKE FELDER LOESCHEN
60 PRINT CHR$(12);
70 VNAME$="":NNAME$="":NR=0:UMSATZ=0
80 KOPF01$=STRING$(57,"-"):FUSS01$=STRING$(64,"-")
90 PRINT KOPF$;"TEST---"
110 SZ%=2:ST%=4:GOSUB 8500:PRINT "TESTPROGRAMM ZUR EINGABE"
120 SZ%=4:ST%=4:GOSUB 8500:PRINT "Vorname"
130 SZ%=4:ST%=40:GOSUB 8500:PRINT "Nachname"
140 SZ%=6:ST%=4:GOSUB 8500:PRINT "Laufende Nummer"
150 SZ%=6:ST%=40:GOSUB 8500:PRINT "Umsatz"
160 SZ%=20:ST%=1:GOSUB 8500:PRINT FUSS01$
170 SZ%=22:ST%=35:GOSUB 8500:PRINT"ESC - Programmende"
180 REM EINGABE LESEN UEBERPRUEFEN
200 SP$=VNAME$
210 SG%=0:SZ%=4:ST%=13:SL=15:STP=2:SE=2:GOSUB 7500
220 VNAME$=SRT$
230 ON SV GOTO 240,60,370,200,330
240 SP$=NNAME$
250 SG%=1:SE=1:SZ%=4:ST%=50:SL=15:STP=2:GOSUB 7500
260 NNAME$=SRT$
270 ON SV GOTO 280,200,370,200,330
280 SP=NR
290 SZ%=6:ST%=30:SL=4:SD=0:STP=1:SE=1:GOSUB 7500
310 NR=SRT
320 ON SV GOTO 330,240,370,220,330
330 SP=UMSATZ
340 SZ%=6:ST%=49:SL=7:SD=2:STP=1:SE=1:GOSUB 7500
350 UMSATZ=SRT
360 ON SV GOTO 200,280,370,200,60
370 END
```

***Bild 2** Testprogramm*

STP Feldtyp (1 numerisch, 2 alphanumerisch)
SP/SP$ Variableninhalt (vom Hauptprogramm zu übergeben)
SE 1 *Muß*-Eingabe, 2 *Kann*-Eingabe
SG% 1 Umwandlung in Großbuchstaben

Nach Rückkehr aus dem Unterprogramm stehen folgende Variablen zur Verfügung:

SRT$ alphanumerische Variable
SRT numerische Variable
SV Feld-/Maskensteuerung: vom Modul unabhängige Verwendung mit ON SV GOTO <ZL1>, <ZL2>, <ZL3>, <ZL4>, <ZL5>,

Als lokale Variablen werden vom Modul verwendet und sollten nicht im Hauptprogamm benutzt werden:

SI, SK%,
SJ% Hilfsvariable
SDEZ$ Komma bereits eingegeben ("E" sonst "")
SU$ Using-Maske für numerische Variable
SR$ eingegebenes Zeichen

Vom Modul werden die angegebenen Steuerzeichen benutzt und über die Rückkehrvariable SRET teilweise dem Bediener zur Verwendung freigestellt:

^S ein Zeichen links (für den Interpreter in ^Z ändern) (siehe Kommentar in Listing Zeile 8015)
^D ein Zeichen rechts

Fortsetzung auf Seite 317

1 MByte Operativspeicher für A 7150

Der A 7150 ist standardmäßig mit zwei OPS-Karten K 3571 mit einer Speicherkapazität von jeweils 256 KByte ausgerüstet. Um Programme abarbeiten zu können, die einen größeren Arbeitsspeicher als 512 KByte erfordern, muß eine weitere OPS-Karte verwendet werden. Der dafür benötigte Steckplatz kann durch Entfernen der ASP-Karte gewonnen werden. Dadurch ist die Anschlußmöglichkeit von peripheren Geräten stark eingeschränkt.
Durch Umbau einer OPS-Karte kann 1 MByte Operativspeicher realisiert werden. Anstelle der 64-KByte-DRAMs werden 256-KByte-DRAMs verwendet. Zusätzliche Hardwareänderungen ergeben sich in der Speicheradressierung und in der Paritätscodierung/-dekodierung. Mit einer OPS-Karte ist somit 1 MByte (abzüglich 128 KByte Bildwiederholspeicher und 32 KByte Firmware) verfügbar. Durch das Betriebssystem werden 640 KByte verwaltet. Ab DCP 3.3 kann durch den virtuellen Diskdriver VIRDRV.SYS der Speicherbereich oberhalb des Bildwiederholspeichers als virtuelles Laufwerk konfiguriert werden (224 KByte). Der freigewordene Steckplatz kann für zusätzliche Nutzerhardware verwendet werden. Eine abgerüstete Variante mit 512 KByte pro OPS-Karte ist möglich.

VEB Mikroelektronik „Karl Marx" Erfurt, Abt. HT3, Rudolfstraße 47, Erfurt, 5010; Tel. 58 28 06

Wunderlich

Schnittstellenanpassung setty(M) für P8000

Das Kommando **setty(M)** ermöglicht den Anschluß spezieller peripherer Geräte über einen seriellen Terminalkanal an den P8000. Dabei entfällt das für den ungeübten Nutzer aufwendige Einbinden neuer Gerätetreiber in den WEGA-Systemkern. Setty(M) benutzt das *Allgemeine Terminal-Interface* tty(4) und stellt die geforderten Schnittstellencharakteristiken wie Bitrate, Parität etc. am WEGA-Terminaltreiber ein. Nach der Initialisierung des entsprechenden Terminalkanals kann der Nutzer aus seiner gewohnten Umgebung mit geeigneten cat(1)-Kommandos die Peripherie bedienen.
Es besteht die Möglichkeit des Erstellens von Initialisierungsmakros, nach denen dann bei einem weiteren setty(M)-Aufruf ein entsprechender Kanal ohne nochmalige Abfrage der Schnittstellencharakteristiken eingestellt wird. Die Nutzerführung erfolgt menügeführt unter Verwendung von Routinen der SCREEN/CURSES-Bibliothek zur Bildschirmarbeit. Setty(M) sollte als universelles, leicht handhabbares Werkzeug für die kurzzeitige Anpassung an spezielle Peripherie verstanden werden.
Das Programm mit der entsprechenden Dokumentation wird zur kostenlosen Nachnutzung angeboten.

VEB Nachrichtenelektronik Greifswald Abt. H/HE, Brandteichstraße 25, Greifswald, 2200; Tel. 6 62 93 (Koll. Philipp)

Rose/Schidlowski

Nutzerführung durch Funktionstastenbelegung mittels Stapelverarbeitung

Eine effektivere Programmierung erfolgt über Funktionstastenbelegungen (wie in MP 3/88, Seite 94, durch ein BASIC-Programm). Da sich die Belegungen entsprechend den Programmen ändern müssen, wird eine Stapelverarbeitung vorgenommen. Die Nutzerführung wurde auf einem CPC 6128 ausgetestet und wird ständig beim Programmieren eingesetzt. Außerdem ist dieses System anwendbar auf anderen Rechnern (z. B. PC 1715 W). Die Tastenbelegungen werden automatisch für Turbo-Pascal, dBase und POWER im Betriebssystem CP/M 3 plus geladen und vor dem Start der Software angezeigt.
Durch den Start des Systems (Kaltstart durch das RSX-Kommando bei CPC) wird **PROFILE.SUB** ausgeführt. Dadurch erfolgt die Tastenbelegung bei CP/M plus. Auch für eine modifizierte Version (80-Spuren-Diskettenlaufwerk, Realisierung der Formate SCP 624 KByte, SCP 780 KByte, CP/A 800 KByte) ist diese Nutzerführung möglich. Bei der Originalversion (Systemdiskette 1 des CPC, 40-Spuren-Laufwerk bei einseitiger Schreibweise) muß erst das Format SCP oder CP/A angewählt, dann SUBMIT PROFILE aufgerufen werden. Das Programmsystem kann natürlich bei Bedarf unter Beachtung des Speicherplatzbedarfs bei der Belegung der Funktionstasten f0 bis f9 erweitert werden. Deshalb sind nicht noch mehr Zeichen programmiert worden. Die Softwarelösung wird kostenlos (per Nachnahme auf 3- oder 5,25-Zoll-Diskette mit Kurzdokumentation) abgegeben.

Dieter Majewski, Roald-Amundsen-Straße 26, Rostock 26, 2520

Nutzerverwaltungssystem LOGON

Das System **LOGON** ermöglicht die Festlegung von Zugriffsrechten für mehrere Nutzer an einem 16-Bit-Rechner mit Festplatte und den Start der ausgewählten Programme. Es ist ein komfortables System, welches Nutzern ohne Betriebssystemkenntnisse erlaubt, mit ihren Anwenderprogrammen zu arbeiten. Viele Servicefunktionen sichern ein bequemes und effektives Arbeiten am Computer. Anwenderprogramme und -daten sind durch den Aufruf von LOGON mittels persönlichen Paßwortes weitestgehend vor dem Zugriff anderer allgemeiner Nutzer geschützt. Jeder Nutzer bekommt nur die Programme angeboten, zu denen er Zugriffsrechte besitzt. Windowtechniken und Selektierung mit den Corsortasten ergeben eine benutzerfreundliche Bedienoberfläche.
Die Installation des Systems auf der Festplatte erfolgt mit Hilfe des Programmes LOGONINST. Beim Start des Computers wird ein Programm zum Stellen der Systemuhr und anschließend LOGON aus der vervollständigten AUTOEXEC.BAT heraus automatisch gestartet, und es können dann nach Eintritt in LOGON durch das persönliche Paßwort die jeweils angezeigten Anwenderprogramme gestartet werden. Zwischen dem Systemmanager und den in Gruppen einteilbaren Nutzern können Nachrichten ausgetauscht werden (MAILBOX auf der Festplatte). Auf einer Textdatei wird automatisch ein Protokoll geführt, welches den Nutzer mit Datum, Anfangs- und Endezeit und aufgerufenen Programmen protokolliert. Ein Systemabbruch wird ebenfalls registriert. Eine monatliche Auswertung nach Gruppen und Abteilungen ergänzt die Servicefunktion.
Es können maximal 100 Anwenderprogramme mit je 100 Anwendern installiert werden. Alle Dateien werden im Programmsystem mittels eines Wordstar-kompatiblen Textprozessors bearbeitet. Auf eine bedienerfreundliche und sichere Menüführung ist besonderer Wert gelegt worden.
Das Programmsystem wurde in Quick-Basic V. 4.0 entwickelt. Hardwarevoraussetzungen sind ein 16-Bit-Rechner mit MS-DOS-kompatiblem Betriebssystem und mit Festplatte sowie ein Drucker.

SDAG WISMUT, Betrieb für Bergbauausrüstung Aue, Direktorat Technik, Abt. TE, Koll. Jähn, Rudolf-Breitscheid-Straße 25-27, Aue, 9400; Tel. 2 42 81/335

Mücke/Ziehe

AutoCad-Zeichnungen auf dem K 6418.02

Für Zeichnungen, die unter AutoCad auf einem IBM-kompatiblen 16-Bit-PC erzeugt wurden, bieten wir Ihnen mit dem Programm **PLK 6418.EXE** eine Softwarelösung für die Ausgabe auf dem Robotron-Plotter K 6418.02 an.

Bauakademie der DDR, Institut für Projektierung und Standardisierung, Abt. COMPAL, Plauener Straße 163/165, Berlin, 1092; Tel. 37 83 24 16

Pialek

Druckertreiber PMODE

Erfahrungsgemäß besteht des öfteren die Notwendigkeit, verschiedene Druckertypen an MS-DOS-Rechnern zu betreiben, die ursprünglich nicht für diesen Anwendungsfall konzipiert waren. Hierbei treten in der Praxis zwei schwerpunktmäßige Probleme auf. Zum ersten die nicht einwandfreie Bedienung der COM-Schnittstellen durch den List-Kanal des DOS (betrifft V.24-Drucker), zum zweiten Zeichensatzprobleme, falls der betreffende Drucker nicht mit einem IBM-Standardzeichensatz ausgerüstet ist.
PMODE wird ähnlich dem MODE-Kommando aufgerufen und erlaubt erweiterte Zuweisungsmöglichkeiten der LPT- und der COM-Schnittstellen untereinander. Die Protokolle werden dabei vollständig angepaßt. Des weiteren kann optional eine Zeichensatzkonvertierung für verschiedene Druckertypen aktiviert werden. Die vorliegende Version läuft auf dem EC 1834, dem A 7150 sowie auf allen XT- und AT-Kompatiblen.

Kombinat VEB Kabelwerk Oberspree „Wilhelm Pieck", Abteilung OM/H, Wilhelminenhofstraße 76-77, Berlin, 1160; Tel. 6 33 26 03

Frentin

Arbeiterbetreuung

Das Programm **Arbeiterbetreuung** unterstützt neben Dateiaufbau und komfortabler Dateipflege umfangreiche Auswertungen, wobei besonderer Wert auf einfache Handhabung und große Variabilität bei der Formulierung der Bedingungen und der Gestaltung/Sortierung der Drucklisten gelegt wurde. Die Länge eines Datensatzes beträgt 81 Byte, wobei Stammnummer, Name und Vorname, Geschlecht, Geburtsdatum, Kostenstelle, Struktureinheit, Datum der anerkannten Betriebszugehörigkeit, Datum des tatsächlichen Eintritts in den Betrieb, sowie bei Unterbrechung des Arbeitsverhältnisses der Grund und der Beginn der Unterbrechung gespeichert werden. Weiterhin besteht die Möglichkeit der Speicherung von ausgezahlten Treueprämien, die alle 5 Jahre gewährt werden.
Durch die im Dialog formulierbaren Auswahlkriterien und Drucklistengestaltungen werden z. B. folgende Auswertungen möglich:
- Betriebsangehörige in einem bestimmten Altersbereich
- Altersstruktur und Druchschnittsalter der Angehörigen des Betriebes/einer Struktureinheit untergliedert in männlich und weiblich
- zu erwartende Betriebszugehörigkeits-/Geburtstagsjubiläen in einem bestimmten Zeitraum
- Betriebsangehörige, deren Arbeitsverhältnis zur Zeit ruht
- Betriebsangehörige einer Kostenstelle/Struktureinheit.

Bei der Arbeit mit Disketten sollte der Datenbestand aus Kapazitäts- und Zeitgründen 3500 Datensätze nicht überschreiten. Das Programm wurde unter Turbo-Pascal für den A 7100/7150 mit SCP 1700 und mit Epson-kompatiblem Drucker erarbeitet.

VEB Kontaktbauelemente und Spezialmaschinenbau Gornsdorf, BfN, Auerbacher Straße, Gornsdorf, 9163; Tel. Meinersdorf 60

Kemnitz

Erika 3004 zur Computersteuerung

Die Schreibmaschine Erika 3004 electronic eignet sich zur Ein- und Ausgabe für die Kleincomputer 85/1 oder 87. Die integrierte Schnittstelle kann dazu direkt genutzt werden. Für den Anschluß sind keinerlei Zusatzmodule notwendig. Die Geräte werden nur durch eine 5polige Verbindungsleitung gekoppelt. Ein spezielles Treiberprogramm im KC (2 KByte) steuert vollständig die Ein- und Ausgabe. Zur Ausgabegestaltung stehen 40 Kommandos zur Verfügung, die mit dem Epson-Drucksteuerzeichensatz kompatibel sind. Vorteile der Lösung sind:
- Die Schreibmaschinentastatur kann parallel zur Rechnertastatur für Eingaben genutzt werden.
- Alle Schreibmaschinenfunktionen können im Online- und im Offline-Modus genutzt werden.
- Im Online-Modus stehen weitere Funktionen zur Verfügung: Fettschrift, Doppelschrift, Sperrschrift, automatische Unterstreichung, komprimierte Schrift, beliebige Zeilenvorschübe.

– Im Offline-Modus existiert ein automatischer Papiereinzug.
– Alle Sondertasten des Rechners sind auf der Schreibmaschinentastatur ebenfalls als solche realisiert, eine Umbelegung ist ohne Schwierigkeiten möglich.
– Die Ausgabe ist sowohl mit deutschem Zeichensatz (Umlaute) als auch mit amerikanischem (Klammern) möglich.
Die Treibersoftware ist als RAM- und als ROM-Version verfügbar, und eine sofortige Nachnutzung ist möglich.

VE Einzelhandelsbetrieb (HO) Dresden, Gaststätten, Abt. Computertechnik, PSF 409, Dresden, 8012; Tel. 475316, App. 15 *Raddatz*

FEM-Postprozessor

Das von uns entwickelte **GRAFEM** ermöglicht eine vollständige grafische Auswertung der Lösung von 2D-FEM-Berechnungen über linearen Dreiecksnetzen auf A 7150, CM 1910, A 7100. Eine Version für den EC 1834 ist in Vorbereitung. Voraussetzung zum Programmeinsatz ist die geladene Firmware GPAFx.xxx. Im Programm erfolgt die Darstellung von Vernetzung, Niveaulinien, Ableitungen und Lösungsgebirge. Letzteres mit Drehung unter Beachtung verdeckter Linien. Mittels einer Lupe lassen sich beliebige Bildausschnitte vergrößern. Obige Darstellungsarten können getrennt oder als Bildserie bearbeitet werden. Die variable Bildbearbeitung erfolgt über eine Menüleiste innerhalb mehrerer Untermenüs. Für die Hardcopy wird ein Nadeldrucker benötigt. Die notwendige Datenübergabe erfolgt als Textfile.

Technische Universität Karl-Marx-Stadt, Sektion Mathematik, Reichenhainer Straße 41, Karl-Marx-Stadt, 9010; Tel. 5612165 *Haase*

Basismodule für Fortran 77

Ziel von **FTNBAS** ist die Bereitstellung einer portablen Schnittstelle für die Entwicklung komfortabler dialogorientierter Fortran-77-Software. Mängel von Fortran bezüglich Tastatureingabe und Bildschirmausgabe werden bei Anwendung der Basismodule beseitigt. FTNBAS unterstützt für die Tastatureingabe die Eingabe von Zeichen, die Eingabe und das Editieren von Zeichenketten sowie die Abfrage und das Löschen des Tastaturpuffers. Für die Bildschirmausgabe sind möglich:
– Einstellen von Bildschirmattributen
– Setzen von Vordergrund- und Hintergrundfarben
– Kursorpositionierung
– Größe des Kursors
– Rollen von Bildschirmausschnitten
– Fenstertechnik
– Pull-down-Menütechnik
– Aufbau von Bildern im Hintergrund.
Weiterhin bietet FTNBAS: Hauptspeichermanipulationen, Systemdienste sowie Routinen zur Dateiarbeit und zur Verzeichnisarbeit.

VEB Datenverarbeitungszentrum Halle, Abteilung FP, Block 081, Halle-Neustadt, 4090; Tel. 6160 *Gruhl*

Handwerkersoftware und Systemsoftware

Wir bieten Anwendersoftware zur Nachnutzung für Handwerksbetriebe im Elektronikbereich und Systemsoftware für den universellen Gebrauch.
Als Anwendersoftware:
– Lohnrechnung, Führung des Kassenbuches
– Schreiben von Reparaturberichten/Rechnungen nach Regelleistung und Stundenlohn
– Führung eines Reparatureingangsbuches und einer Kundendatei
– Schreiben von Belegen: Lieferscheine, Post- und Bahnbelege unter Verwendung der Dateien.
Als Systemsoftware:
– Diagnose- und Dienstprogramme zur Prüfung und zur Optimierung von Festplatten
– Konsolenemulation zur Abarbeitung von CP/M-Programmen auf 16-Bit-MS-DOS-Systemen
– Umleitung der parallelen Druckerschnittstelle in ein Datenfile und Druck-Dienstprogramme
– Dateieditor für dBase III-Dateien
– EPROM-Programmiergerät (2716 ... 27256)
– Analyseprogramm für die Integrität von Programmdateien (Feststellung von Computerviren)
Hardwarevoraussetzungen: IBM-kompatibler 16-Bit-PC mit mindestens 512 KByte RAM, 1 Diskettenlaufwerk (besser Festplatte), Matrixdrucker.

Werkstatt für elektronische und technisch-physikalische Anlagen, Straße der DSF 22, Templin, 2090; Tel. 2929 *Friedrich*

Schneider-PC am Farbfernseher

Mit der Verbreitung der KC-Technik wurden viele Hörsäle und Unterrichtsräume mit Farbmonitortechnik ausgestattet. Um diese Technik für 16-Bit-Rechner nutzen zu können, wurde eine Leiterplatte entwickelt. Damit können Schneider-PCs an die im Hörsaal installierten Farbfernsehgeräte (4000er Serie Colortron, Colormat) angeschlossen werden. Die Leiterplatte, welche zwischen PC-Monitor und Rechner gesteckt wird, wandelt die digitalen Signale um, so daß sie dem RGB-Eingang der Farbfernsehgeräte zugeführt werden können. Es werden alle 16 Farben in hoher Qualität dargestellt. Prinzipiell ist jeder Farbfernseher geeignet, der über einen analogen RGB-Eingang verfügt. Die o. g. Farbfernsehgeräte der DDR-Produktion sind werksseitig für den RGB-Eingang vorgesehen, jede Fernsehgerätewerkstatt kann diesen geringfügigen Umbau vornehmen. Auch die Kofferfarbfernsehgeräte der DDR-Produktion sind gleichermaßen geeignet. Der Anschluß eines SW-Monitors oder Fernsehgerätes mit Videoeingang (BAS) ist ebenfalls möglich, die Farben werden in 16 Graustufen dargestellt.
Bei hinreichender Nachfrage kann die komplette Zusatzleiterplatte von uns bezogen werden.

Technische Universität „Otto von Guericke" Magdeburg, BfN, Koll. Iffarth, PSF 124, Magdeburg, 3010; Tel. 5920 *Prof. Dr. Lorenz*

Dateiübertragungssystem zwischen KC und PC

Oft besteht die Aufgabe, für den KC 85 entwickelte Software (Basic- oder Pascal-Programm) auf andere Rechner übertragen zu müssen. Auch die Übertragung auf Diskette vorliegender Programme oder Daten zum KC 85 ist wünschenswert. Um diesen Aufgaben gerecht zu werden, wurde an der TU Magdeburg das Programmsystem **DFX** entwickelt, womit die Übertragung einer Kassettendatei zwischen dem KC und einem Rechner vom Typ PC 1715, A 51xx oder A7100 möglich ist. Das Programmsystem DFX besteht aus dem von der zu übertragenden Datei unabhängigen Programm DFX und den Konvertierungsroutinen für entsprechende Programmiersprachen oder Textverarbeitungssysteme. Konvertierungsroutinen sind derzeit für Pascal- und Basic-Quelltext sowie für Datenfelder und für das Textverarbeitungssystem WordPro implementiert.

Technische Universität „Otto von Guericke" Magdeburg, Sektion Informatik, WB-MI, PF 124, Magdeburg, 3010; Tel. 592766 *Hinz*

Programmpaket für Handwerksbetriebe

Gewerke: Dachklempnerei, Sanitärinstallation, Heizungsinstallation
Anpassung: Elektroinstallation, Dachdecker, Zündungs- und Vergaserdienst, Fotograf
Hardware: IBM-PC/XT/AT-kompatibler Computer mit mindestens einem Laufwerk und 512 KByte RAM, z. B. Schneider PC 1512/1640, EuroPC. Commodore AMIGA und ATARI-ST sind nur mit MS-DOS-Emulator nutzbar.
Programme:
1. Schreiben von Handwerksrechnungen
2. Erstellen von Monats- und Jahresbilanzen (PA-Liste, HQ-Bogen ...)
3. Lohnabrechnung
4. Kassenbuch mit Anlagenverzeichnis und Jahresabschluß
5. Materialkartei Einkauf, Inventur und Arbeitsmittelverzeichnis
6. Serviceprogramm zur Dateipflege

Die Programme können einzeln genutzt werden. Bei Benutzung aller Programme kann jederzeit innerhalb weniger Sekunden ein kompletter Jahresabschluß durchgeführt werden. Zum Programmpaket wird eine 120 KByte lange Dokumentation geliefert.

Firma Roland Fuchs, Dachklempnerei und Sanitärinstallation, Ernst-Thälmann-Straße 48, Prenzlau, 2130; Tel. 2038 *Fuchs*

Leistungsverzeichnis und Rohrnetzberechnung

Hardware: IBM-PC/XT/AT-kompatibler Computer mit mindestens einem Laufwerk und 512 KByte RAM, z. B. Schneider PC 1512.
Programm Leistungsverzeichnis:
– Erstellen von Leistungsverzeichnissen nach PVK und PL.60 sowie von Materiallisten.
– Gewerke Heizungsinstallation, Sanitärinstallation, Lüftungsinstallation; weitere Gewerke können selbst aktiviert werden.
– Speichern und Editieren der Leistungsverzeichnisse
– Aufbau und Pflege der Preisdateien
– Preisdateien Heizung und Sanitär im Lieferumfang enthalten
– auf Wunsch Änderungsdienst.
Programm Rohrnetz:
– Berechnung von 2-Rohr-Heizungsrohrnetzen
– Verästelung oder Tichelmann
– Ausdruck der Rohrnetztabelle
– Darstellung und Ausdruck des grafischen Druckverlaufes
– Speichern und Editieren der Rohrnetze
– eigene Rohrbibliothek.

Die Programme besitzen eine benutzerfreundliche Oberfläche und sind durch Anwendung der Sprache C kompakt und schnell. Auf Wunsch kann eine Demo-Version der Programme zur Verfügung gestellt werden.

Firma Roland Fuchs, Dachklempnerei und Sanitärinstallation, Ernst-Thälmann-Straße 48, Prenzlau, 2130; Tel. 2038 *Fuchs*

Papiersparender Druck von Textdateien

Das vorliegende TURBO-PASCAL-Programm druckt Textdateien in papiersparenden Formaten. Es werden dabei Schriftart, Zeilenabstand und Seitenvorschub verändert. Eine Druckseite enthält 85 Zeilen; also 20 Zeilen mehr als bei Standard-Druckfunktionen. Der Anwender kann auch noch 150 Zeilen je Seite auswählen. Bei diesem Druckformat ist ein guter Drucker und ein gutes Farbband Voraussetzung. Die Seitenlänge beträgt immer 12". Auf Wunsch kann man Endlospapier beidseitig bedrucken oder Einzelblatt-Verarbeitung benutzen. Der Druck von ausgewählten Textteilen wird über die Seitennummer gesteuert.
Das menügesteuerte Programm ist unter dem Betriebssystem DCP 3.20 auf dem EC 1834 entwickelt und getestet worden.

Zentrales Forschungsinstitut des Verkehrswesens der DDR, Zentrum für Personenverkehr und Verkehrsnetze, PSF 403, Berlin, 1017; Tel. 4922875 *Neumann*

Wir suchen ...

... eine Softwarelösung für die Dokumentation eines Milchkuhbestandes unter SCP/CPA.

LPG Milchproduktion, Jüdendorf, 4241; Tel. Nebra, 5200 *Philipp*

Fortsetzung von Seite 315

^H		Backspace
^T		Feld löschen

im Beispiel verwendet für

^A	SRET=2	Kursor ein Feld zurück
^E	SRET=4	Kursor in das erste Feld auf dem jeweiligen Bild
^X	SRET=5	Kursor in das letzte Feld
ESC	SRET=3	Verlassen des Teilprogramms
ET,^F	SRET=1	Abschluß der Eingabe, Kursor in das nächste Feld

Helmut Schenk



Produktpalette von DEC erweitert

Nach dem Aufgeben eines eigenen RISC-Projektes von 1987/88 kündigte die Firma Digital Equipment Corporation (DEC) Anfang des Jahres mit der DECstation 3100 ein System auf Basis der RISC-Prozessoren R 2000 und R 2010 der Firma MIPS Computer Systems an, das dem Workstationbereich zuzuordnen ist. Es umfaßt die Modelle **DECstation 3100**, **DECstation 3100S** und das **DECsystem 3100**. Über diese Reihe berichteten wir bereits ausführlicher in MP 9/89, 4. Umschlagseite. Kürzlich wurde, ebenfalls auf Basis der R 2000-CPU, als RISC-Einstiegsmodell mit etwa 73 Prozent der Verarbeitungsleistung einer DECstation 3100 das Modell **DECstation 2100** angekündigt. Bei einem Preis von etwa 20000 DM soll diese Workstation mit einer Leistung von 10 MIPS somit billiger sein als einige Personalcomputer mit dem Prozessor 80386. Die DECstation 2100 hat einen Hauptspeicher mit 8 bis 24 MByte sowie wahlweise einen 15- oder 19-Zoll-Monitor.

Als weitere RISC-Systeme wurden darüber hinaus Modelle mit dem leistungsfähigeren Prozessor R 3000 angekündigt:

Das **DECsystem 5400** als Mehrnutzer- und Steuersystem im 40-Zoll-Standgehäuse. Es hat bis zu 64 MByte RAM und einen DSSI-Festplattenadapter für bis zu 9,7 GByte Massenspeicher.

Das **DECsystem 5810** soll vor allem als Fileserver eingesetzt werden. Dazu dienen die Leistung von etwa 19 MIPS, 2,2 GByte Plattenspeicherkapazität, 32 MByte RAM sowie ein zweistufiger Cachespeicher.

Im **DECsystem 5820** werden statt einem zwei R 3000-Prozessoren eingesetzt, auch die Kapazität des RAMs wurde verdoppelt. Als Leistung werden 36 MIPS angegeben. Bei beiden Versionen lassen sich über den VAXBI-Bus der lokale Hauptspeicher auf 256 MByte und die Plattenkapazität bis zu 115 GByte erweitern.

Auch bei der mit VAX-Prozessoren ausgestatteten Modellreihe wurden von DEC im Laufe des Jahres Veränderungen vorgenommen und folgende Systeme angekündigt:

- **VAXstation 3100** mit 3 bis 4 MIPS
- **VAXstation 3520** mit 2 Prozessoren und der 5fachen Leistung einer VAX 11/780
- **VAXstation 3540** mit 4 Prozessoren und der 10fachen Leistung einer VAX 11/780
- **MicroVAX 3800** als Ersatz der MicroVAX 3500
- **MicroVAX 3900** als Ersatz der MicroVAX 3600 und leistungsfähigstes Modell der MicroVAX-Serie
- **VAXserver 3800** und **VAXserver 3900** speziell für den Einsatz in lokalen Netzen.

Bemerkenswert ist auch, daß DEC nunmehr selbst PCs anbietet – in den USA nach DEC-Spezifikationen von Tandy gefertigt. Dabei handelt es sich um die Modelle

- **DECstation 210** mit 80286 und 10 MHz
- **DECstation 316** mit 80386 und 16 MHz
- **DECstation 320** mit 80386 und 20 MHz

Als einer der ersten Hersteller bietet die Firma mit DECwindows eine gemeinsame Benutzeroberfläche für das eigene VMS-Betriebssystem sowie für Ultrix und MS-DOS an. *MP*

Neue PCs von IBM



Mit zwei neuen Modellen hat die Firma IBM ihre Produktpalette erweitert.

Nach dem Mißerfolg mit ihrem ersten tragbaren PC, dem Convertible (auch AP genannt), der sich gegen die Konkurrenz nicht durchsetzen konnte, wurde jetzt ein neuer Anlauf genommen. Mit seinen 9,5 kg Gewicht zählt der **P70 386** nicht zur Klasse der weit verbreiteten Laptops, sondern ist erklärtermaßen ein „full function movable", das heißt ein voll funktionsfähiger mobiler PC-Arbeitsplatz, der über dieselben Funktionen wie ein Tischgerät verfügt. Charakteristisch ist die Mikrokanalarchitektur, so daß der P70 386 das mobile Modell des PS/2 darstellt. Bereits in der Grundausstattung verfügt er über 4 MByte Arbeitsspeicher, der sich auf der Hauptplatine auf 8 MByte aufrüsten läßt. Da 85-ns-RAM-Bausteine verwendet werden, kann die CPU mit Wartezyklen zwischen 0 und 2 arbeiten. Der 10-Zoll-Plasmabildschirm hat eine Auflösung von 640 × 480 Punkten und kann in den Modi CGA, EGA und VGA betrieben werden, wobei die Farben in bis zu 16 Graustufen darstellbar sind. Der P70 386 wird in zwei Varianten geliefert: Das Modell K61 hat eine 60-MByte-Festplatte, während das Modell K21 eine 120-MByte-Festplatte besitzt.

Als zweite Erweiterung des PS/2 präsentierte IBM das **Modell 55 SX**, mit dem die Firma nunmehr, wie schon viele Hersteller, den Prozessor 80386 SX als Einstieg in die 32-Bit-Klasse nutzt. Der 16-MHz-Prozessor ermöglicht diesem Mikrokanalrechner eine um etwa 20 Prozent höhere Geschwindigkeit als sie vergleichbaren 80286-Systemen möglich ist. In dem kompakten Gehäuse des Modells 30 bietet der 55 SX drei 16-Bit-Erweiterungssteckplätze. Als Speicher stehen u. a. 2 MByte RAM sowie 30- oder 60-MByte-Festplatten zur Verfügung. *MP*

Weitek-Koprozessor für Intel 80486

Mit dem Integrieren der bisher separaten mathematischen Koprozessoren auf Mikroprozessorchips – so beim Intel 80486 und beim Motorola 68040 praktiziert – sahen viele das Aus für diese Zusatzprozessoren gekommen. Die Integration bietet vor allem den Vorteil der kurzen Signalwege zwischen Mikroprozessor und Numerikprozessor.

Die für ihre schnellen Numerikprozessoren bekannte Firma Weitek gab dennoch nicht auf und entwickelte ihren von 386er Systemen bekannten Abacus 3167 zum 4167 weiter. Der Aufwand scheint sich gelohnt zu haben: Weitek gab bekannt, daß rechenintensive wissenschaftliche und technische Anwendungsprogramme zwei- bis dreimal schneller beim Einsatz des 4167 ablaufen als mit der On-chip-Gleitkommaeinheit des 80486. Da der 4167 mit Memory-mapped-Protokoll arbeitet, erhält er die Befehle über den Adreßbus und die Daten über den Datenbus, so daß er weniger Taktzyklen benötigt im Vergleich zu Numerikprozessoren, die Befehle und Daten über *einen* Bus erhalten. Außerdem enthält der 4167 sechzehn 64-Bit-Register (doppelt so viel wie die Fließkommaeinheit des 80486), so daß er mehr Berechnungen ohne Datentransporte zwischen externem Speicher und Gleitkommaeinheit ausführen kann. So soll der Gleitkomma-Datenpfad des 4167 bis zu 500 Prozent schneller sein als der Datenpfad der 80486-Gleitkommaeinheit. Der 4167 ist in 1,2-µm-CMOS-Technik hergestellt; für den Einsatz wird er in einen 142poligen PGA-Sockel auf der Systemplatine gesteckt. *MP*

Mobile Datenbestände auf Festplatten

Die Firma Kyocera Electronics Europe stellte im Frühjahr ebenfalls ein Konzept portabler Datenbestände auf Festplatten unter dem Namen KT 20/30 vor (das Prinzip hatten wir bereits in MP 11/88 anhand der Drive Box erläutert). Das Bild zeigt sehr gut, wie ein spezieller Rahmeneinschub mit den Maßen einer 5 1/4-Zoll-Geräteöffnung ein 3,5-Zoll-Festplattenlaufwerk aufnimmt. Während der Rahmen im XT- oder AT-kompatiblen PC fest installiert wird, ist das Laufwerk in ein steckbares und damit leicht wechselbares Gehäuse eingebettet. Die Kapazität der somit als persönliche Datenpacks verwendbaren Festplattenlaufwerke liegt bei 20 bzw. 30 MByte. *MP*

Zeichnungsverwaltung für CADdy

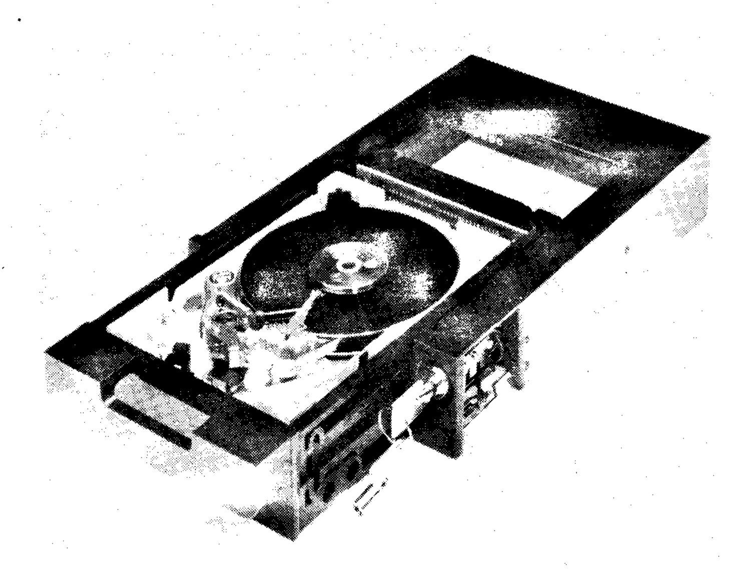
Auf der CAT'89 stellte ZIEGLER-Instruments ein neuentwickeltes Zeichnungsverwaltungssystem für seine CAD-Software CADdy vor: **CADdy ZV**. Das datenbankorientierte Informationssystem dient der übersichtlichen Verwaltung vorhandener Zeichnungen, beschleunigt das Auffinden von Bildern und entlastet von Arbeiten auf Betriebssystemebene. CADdy ZV arbeitet mit der Maskentechnik, die dem Anwender von CADdy her vertraut ist; die Einbindung in das CAD-Kernsystem schafft direkte Zugriffsmöglichkeiten von der Datenbank auf Zeichnungen und umgekehrt. Die Suche nach einzelnen Zeichnungen erfolgt wahlweise über einzelne Merkmale oder Merkmalskombinationen. Neben Suchbegriffen lassen sich auch Bereichsbedingungen (z. B. „Datum von ... bis") definieren. Die gesuchten Datensätze werden unmittelbar angezeigt: Ohne die Zeichnungsverwaltung zu verlassen, kann die dazugehörige Zeichnung direkt auf dem Grafikbildschirm eingeblendet werden.

Mit CADdy ZV befreit sich der Anwender von den Beschränkungen durch das Betriebssystem MS-DOS. Die Dateiorganisation wird vereinfacht und erweitert; vor allem aber wird eine aussagekräftige Benennung von Zeichnungen möglich, da sich der Anwender nicht mehr an den üblichen achtstelligen MS-DOS-Namen halten muß. *MP*

Turbo-Datenbank

Informix-Turbo und Informix-Net sind neue Softwareprodukte der Datenbankfamilie Informix, die Siemens seit Juli für seine Sinix-Rechnersysteme anbietet.

Mit *Informix-Turbo* läßt sich die Leistung bisheriger Informix-Datenbanken durchschnittlich um den Faktor 2 steigern. Damit gehören sie nach Aussage des Herstellers zu den derzeit schnellsten Datenbanken, die auf Unix-Rechnern ablaufen. Diese Leistungsfähigkeit braucht man z. B. bei größeren Rechnern mit mehr ange-



schlossenen Bildschirmen, bei Datenbanken mit großem Datenvolumen oder bei Anwendungssystemen mit hohen Transaktionsraten.
Die neue Software *Informix-Net* ist die Netzwerk-Komponente der Informix-Produktfamilie. Sie ermöglicht den Zugriff auf Datenbanken, die auf einem anderen, fernen Rechner liegen. Informix-Net funktioniert nach dem Client-Server-Prinzip, das im Unterschied zu echt verteilter Verarbeitung keine großen organisatorischen Probleme aufwirft. Die Anwendungsprogramme laufen in den Arbeitsplatzrechnern (Clients), die Datenbank ist als logische Einheit auf dem Server-Rechner. Jeder Rechner, auf dem Informix-Net installiert ist, kann sowohl als Client wie als Server fungieren. Dadurch lassen sich mehrere Datenbanken innerhalb eines Netzes verteilen, wobei von jedem Rechner auf jede Datenbank zugegriffen werden kann. *MP*

Kupfer-Plast-Verbindung mit doppelter Festigkeit

Die Firma General Electric entwikkelte ein Verfahren, mit dem es möglich wird, Metalle (Kupfer) mit Plast zu verbinden. Diese Verbindung könnte in der galvanischen Verkupferung von technisch bearbeiteten, elektronischen Schaltkreisplatten aus Plast angewendet werden.
Diese als sehr fest charakterisierte Kupfer-Plast-Verbindung wird mit Hilfe des Metallspritzverfahrens hergestellt und weist eine Ablösefestigkeit auf, die fast doppelt so groß ist wie bei herkömmlichen Verfahren, bei denen das Kupfer mechanisch aufgetragen wird und bei denen durch den Ätzvorgang mikroskopische Risse im Plast hervorgerufen werden.

Quelle: New Technology Week vom 3. 4. 1989 *Fa*

Bauelemente mit Quantenstrukturen

Von Wissenschaftlern der Firma Texas Instruments sollen Transistorfunktionen realisiert worden sein, die auf Quantendimensionen begrenzt sind und die auf Quanteneffekten beruhen. Gegenüber bereits früher angekündigten Bauelementen mit quantenelektrischen Eigenschaften wurden in Größe und Funktion neue Dimensionen erreicht. Der Bipolar-Resonanz-Tunnel-Transistor nutzt das Verhalten von Materie und Energie bei Strukturgrößen unter 0,02 μm aus.
Die Quanten-Bauelemente sind bisher nur im Labor realisiert worden. Bis zur praktischen Anwendung könnte aber noch ein Jahrzehnt vergehen.
Mit den Quanten-Transistoren könnte eine Revolution in der Halbleitertechnologie ausgelöst werden. Die aktiven Elemente sind 100mal kleiner als die entsprechenden Funktionselemente derzeitiger Halbleiter. Die erreichbaren Schaltzeiten sollen um das 1 000fache höher liegen.
Der Resonanz-Tunnel-Transistor ist eine Etappe auf dem Weg zum unipolaren Quanten-Transistor. Die bipolaren Transistoren in Gallium-Arsenid, Aluminium-Gallium-Arsenid und Indium-Gallium-Arsenid zeigen jedoch schon eine vielversprechende Leistung. Bei Raumtemperatur werden Stromverstärkungen um den Faktor 50, in Extremfällen bis 450, erreicht. Um derartige Strukturen in Serie produzieren zu können, sind neue Schaltkreis-Architekturen und entsprechende technologische Ausrüstungen noch zu entwickeln.

Quelle: eee. Elektronik-Technologie. – Leinfelden-Echterdingen (1989) 6. – S. 42–43 *Wi*

Bondbare Schicht für einen keramischen Chip-Carrier

Chip-Carriers bilden die Verbindung zwischen einem IC-Baustein und der übergeordneten Verdrahtungseinheit. Besteht dieser Chip-Carrier aus keramischen Werkstoffen, so trägt der Isolationswerkstoff, z. B. Aluminiumoxidkeramik, eine Sintermetallschicht, die aus Molybdän/Mangan bzw. Wolfram bestehen kann und die zu einem Leiterbild strukturiert wird. Damit wird die Voraussetzung geschaffen, um einen elektrischen Kontakt zwischen dem IC-Baustein und der den Chip-Carrier umgebenden Schaltung zu ermöglichen.
Der Kontakt zwischen den Anschlußpads des ICs und der Metallisierung des Carriers wird in der Regel durch Drahtbonden mit Gold- bzw. Aluminiumdraht bis herab zu 20 μm Durchmesser hergestellt. Da die Sintermetallschichten nicht bondbar sind, ist ein Weitermetallisieren z. B. mit Nikkel und Gold erforderlich. Dazu nutzt man elektronische Verfahren. Bevorzugt werden außenstromlose Prozesse. Damit lassen sich Kombinationsschichten aus Nickel/Bor mit 1 % Boranteil und Feingold in einer Gesamtdicke von 4 bis 5 μm aufbringen.
Die Firma Siemens hat für die keramische Schichtplatte mit Chip-Carrier-Funktion eines optoelektronischen Wandlers einen außenstromlosen Metallabscheidungsprozeß entwikkelt, der reproduzierbare Haftfestigkeitswerte von etwa 5 cN eines Golddrahtes mit 22 μm Durchmesser, Dehnung 2 bis 4 %, auf eine NiB/Gold-Schicht bringt. Die Bondverbindung wird mittels des Thermosonic-Nailhead-Bondens geschaffen.

Quelle: Technische Rundschau. – Bern (1989) 9 *Fa*

Bildchip mit mehr als 4 Mio Elementen

In den Forschungslabors der Firma Kodak wurde ein CCD-Bildchip mit mehr als 4 Mio Elementen entwickelt. Im Gegensatz dazu verfügen bisherige japanische Video-Kameras und Still-Video-Fotoapparate über Bildchips mit maximal 400 000 Bildelementen.
Bereits vor zwei Jahren gelang es der Entwicklergruppe bei Kodak, einen ladungsgekoppelten Sensorchip (CCD-Chip) mit 1,4 Mio Elementen zu entwickeln. Dieser Chip wird inzwischen in der von der Kodak-Tochterfirma Videk gebauten Megaplus-Kamera verwendet, die als die derzeit höchstauflösendste CCD-Kamera bezeichnet wird. Ihr Einsatz erfolgt vorwiegend in der Forschung.
Der Chip besteht aus 2 048 × 2 048 Elementen auf einem Quadrat von 18,4 mm Kantenlänge. Das bedeutet, jeder der rund 4,2 Mio Bildpunkte hat eine Kantenlänge von 9 μm. Die Pixel sind fugenlos aneinandergereiht, und damit ist die gesamte Fläche des Chips sensitiv.
Die gegenwärtig erreichte Aufnahmegeschwindigkeit beträgt 5 Bilder/Sekunde, wobei während dieses Zeitraumes 20 Mio Pixel-Informationen verarbeitet werden müssen. Für das Zwischenspeichern eines einzigen Bildes ist eine Speicherkapazität von 4 MByte erforderlich.
Über den Einsatz dieses Bildchips wurden von Kodak noch keine näheren Angaben veröffentlicht.
In Japan sollen noch in diesem Jahr Video-Kameras mit 500 000 Pixel auf den Markt kommen, und in einem weiteren Entwicklungsschritt wird die 1-Mio-Grenze angestrebt.

Qelle: Die Welt vom 25. 4. 1989 *Fa*

Optische Speicher mit Phasenwechsel

Die Firma Matsushita setzte erstmals das Phasenwechsel-Verfahren zur Datenspeicherung auf optischen Speichern ein. Dazu stellte sie eine 3,5-Zoll-Optical-Disk nebst Laufwerk mit einer Speicherkapazität von 280 MByte und einer Zugriffszeit von 42 ms vor.
Im Gegensatz zum TMO-(Thermo-Magnetical-Optical-)Verfahren wird bei dem Phasenwechsel-Verfahren die Datenspeicherung durch Erhitzen mit Laserlicht bewirkt.
Bis 1990 will die Firma eine 5,25-Zoll-Platte mit einer Speicherkapazität von 640 MByte auf den Markt bringen.
Entwickelt wurde das Phasenwechsel-Verfahren von der Firma Energy Conversion Devices. *Fa*

Vakuum-Mikroelektronik schaltet extrem schnell

An der Entwicklung kleinerer und schnellerer elektronischer Schaltkreise arbeiten die Wissenschaftler des Electrical Technical Laboratory in Tsukuba, Japan. Den Wissenschaftlern ist es gelungen, ein Vakuum-Mikroelektronik-Bauteil zu entwickeln, das als Voraussetzung für die Produktion ultrafeiner Bildschirme gilt. Dabei rechnet man damit, daß diese Bauteile sowohl für das kommende hochauflösende Fernsehen (HDTV), für noch schnellere Supercomputer, als auch für hochentwickelte Telekommunikations-Ausrüstungen in den neunziger Jahren eingesetzt werden können.
Das Bauteil ist 7 μm lang und hat einen Durchmesser von 8 μm. Damit sind diese Bauteile wesentlich kleiner als ihre Vorgänger. Werden diese Elemente auf einer Siliziumscheibe zusammengesetzt, arbeiten sie mit einer sehr viel geringeren Stromstärke als Halbleiter und bringen sehr viel höhere Frequenzsignale hervor, die gleichzeitig weniger anfällig gegen Temperaturschwankungen und Strahlung sind.
Nach Angaben der Wissenschaftler sollen An- und Abschaltzeiten von 1 Trillionstel Sekunde erreicht werden. Das Element reagiert auf Licht und nicht mehr auf Elektronen. Die Datenübertragung soll über Glasfaser erfolgen. Die Geschwindigkeit der Mikro-Bauteile soll dem Zehnfachen der Schaltzeiten der heutigen schnellsten elektronischen Geräte entsprechen.
Um dies erreichen zu können, müssen die Wissenschaftler aber noch die von diesen Geräten ausgehenden Stromemissionen verstärken.
Die japanischen Wissenschaftler nutzen bei ihrer Arbeit die heutige Standard-Chip-Technik, um die winzigen Vakuum-Kathoden herzustellen. Sie bedeckten die Siliziumscheiben mit einer feinen Schicht von Wolfram und Siliziumnitrit. Die Siliziumätzung erreichte man mit einem photolithographischen Prozeß und schuf dabei sich zuspitzende Pyramiden auf dem Material. Um diese Pyramiden herum wurden isolierende Schichten von Wolfram, Tantalum und Silikonoxid gelegt. Sind die Pyramiden einem elektrischen Feld ausgesetzt, so erzeugen sie dann einen Elektronenstrahl. Dabei sind für diese Elemente nach Angaben der japanischen Wissenschaftler keine supraleitfähigen Materialien nötig, vielmehr ist der Ausstoß von Elektronen wichtig. Die Leistung sei um so besser, je kleiner die Geräte sind. Dadurch sei weniger Strom erforderlich, um ein elektrisches Feld aufzubauen und die Elektronen zu emittieren.

Quelle: Blick durch die Wirtschaft vom 5. 5. 1989 *Fa*

Flachdisplay mit großer Helligkeit

Die japanische Firma Meitaku System Co., Ltd., entwickelte ein Flachdisplay, das eine viermal höhere Helligkeit aufweisen soll als vergleichbare Produkte. Dieses Display wird unter der Bezeichnung L POP angeboten.
Die Rückplatte des Displays ist aus Acryl-Kunststoff und wurde durch eine spezielle Drucktechnik mit einer irregulär reflektierenden Oberfläche ausgestattet. An der Seite der Platine kann Licht eintreten, das an der entsprechend gestalteten Oberfläche irregulär reflektiert wird. Dadurch wird eine hohe und gleichmäßige Ausleuchtung des gesamten Paneels ermöglicht.
Die Plattenstärke variiert dem Einsatz entsprechend zwischen 6 und 24 mm. Die Vorzugsfläche ist A4, wobei es sowohl kleine Platten zur Rückbeleuchtung von Flüssigkristallanzeigen als auch große mit Abmessungen bis zu 1 200 × 2 400 mm gibt.
Das gesamte Display hat eine Stärke von 40 bis 55 mm und kann als energiesparende Leuchtfläche eingesetzt werden.

Quelle: New Technology Japan (1989)3 *Fa*

Hardcopy-Bilder ab TV

In Japan wurde ein Verfahren entwikkelt, mit dem es möglich wird, TV-Magnetbandaufzeichnungen als Papierbilder oder Dias herzustellen. Dabei soll die Qualität denen der üblichen Fotos entsprechen. Bei bisherigen Hardcopy-Einzelbildern bestand der Nachteil in der Zeilenstruktur. Bei dem jetzt entwickelten Verfahren wird die Zeilenzahl künstlich vervierfacht. Dabei werden zwischen zwei nebeneinander liegenden TV-Bildzeilen drei zusätzliche Zeilen durch elektronische Interpolation erzeugt und in das Gesamtbild eingefügt.

Quelle: Elektronik. – München 38 (1989) 9. – S. 7 *Fa*

Ingenieure in der DDR

Soziologische Studien, Schriftreihe Soziologie, Autorenkollektiv, Ltg. M. Lötsch, Dietz Verlag Berlin 1988, 200 S., 7 Abb., 8 Tab., DDR 6,20 M

Unter dem Rahmentitel „Ingenieure in der DDR" sind relativ selbständige Themen von fünf Autoren zusammengefaßt. Die Schwerpunktfragen beziehen sich auf Funktion, Verantwortung, Qualifikation, Leistung, Leistungsmotivation und Leistungsanerkennung des Ingenieurs im gesamtgesellschaftlichen Reproduktionsprozeß. Fragen der Denk- und Lebensweise, der inneren Triebkräfte zur Herausbildung und Selbstverwirklichung der Ingenieurpersönlichkeit, seiner geistigen Kultur und seines spezifischen Arbeitsstils sind in dieser Schrift nicht erfaßt.[1] Die soziologischen Studien in der sehr informativen und kompakt geschriebenen Abhandlung beziehen sich nahezu vollständig auf die Frage: Was ist die Funktion des Ingenieurs, was muß er leisten, damit er den an ihn gestellten Anforderungen gerecht wird? *Noch nie war das Gewicht der ingenieurtechnischen Leistungen für ökonomischen und sozialen Fortschritt in unserem Land so hoch wie in der gegenwärtigen Etappe der wissenschaftlich-technischen Revolution* (S. 168, Adler). Und die wesentlichste Schlußfolgerung lautet: *An Bedeutung gewinnen all jene Konsequenzen des Leistungsprinzips, die unabdingbar sind, um die sozialen und personellen Voraussetzungen für das Hervorbringen wissenschaftlich-technischer Spitzenleistungen zu verbessern: ...* (S. 166, Adler).
Aber: *Innovationsprozesse sind unbequem* (S. 18, Lötsch)! Wie das? In der Tat entsteht genau hier der Konflikt, dem sich der Ingenieur tagtäglich gegenübergestellt sieht, wenn er seine soziale Funktion erfüllen will. Deshalb gilt es, *die spezifische Überlegenheit des Sozialismus schon heute zu prägen: vor allem bei der Umsetzung des wissenschaftlich-technischen Fortschritts in sozialen Fortschritt.* Was aber sind denn *jene Konsequenzen des Leistungsprinzips, die unabdingbar sind,* um den Ingenieur zu befähigen, ja zu ermächtigen, dieses Ziel, den sozialen Fortschritt zu gewährleisten und durch höchste Produktivität auch tatsächlich erfüllen zu können? Unsere Soziologen befragen, um dies zu ergründen, den *sozialistischen Ingenieur.* Kennt er die Antwort, da es ihm an höchster Produktivität noch mangelt?
Dort, wo der Ingenieur mit seinen Antworten der Erwartung der Gesellschaft entspricht, werden Soziologen durch den Ingenieur bestätigt, dort, wo er aber eckig, kantig, eigenwillig und nicht selten gerade dadurch originär und hochproduktiv ist, kollidiert er nicht selten mit den üblichen Erwartungen. Es ermangelt gerade dort der Originalität von Ingenieurleistungen, wo eine *mustergültige Geschäftigkeit* anzutreffen ist.
Die Schrift „Ingenieure in der DDR" tangiert diese schwerwiegenden Fragen, aber erschließt nicht konsequente Antworten. *Genau an dieser Stelle beginnt das Problem,* sagt Lötsch: Die Besonderheiten der Interessen des Ingenieurs, der Wertorientierungen und des Verhaltens, seiner Arbeitsbedingungen und seiner Lebensweise sind wohl entscheidend!
Zu diesen Zusammenhängen kann man viel, ja Mustergültiges von den sozialistischen Ingenieurpersönlichkeiten G. Jäger, J. Wernstedt und D. Engelage, die in der Broschüre knapp vorgestellt werden, erfahren (S. 82, Römer). Aber auch Dagmar Hülsenberg, Präsidentin unseres Ingenieurverbandes, leistet als Frau – im Kontrast zu den in der Schrift erörterten anderen Beispielen – geradezu Ungewöhnliches (S. 117, Bernhardt).
Der Ingenieur ist der unmittelbare Schöpfer der gesellschaftlichen Produktivität und der Entwickler unserer neuen technischen Produkte. Die meisten Ingenieure in der DDR wünschen sich eine gesellschaftliche Wertung und Wertschätzung, die der der Leistungssportler nicht nachstehen sollte. Die besprochene Schrift stützt diese Einsicht, sie belegt faktenreich und sachkundig notwendige gesellschaftliche Anforderungen dazu. Sie ist auch gut lesbar, wozu der Wechsel des Stils der verschiedenen Autoren sogar beiträgt.

[1] Der Rezensent empfiehlt eine solche Schrift unter Leitung von A. Erck, Technische Hochschule Ilmenau.

Prof. Dr. Dr. M. Roth

Computertechnik

von K.-P. Scholz (Hrsg.), Lexikon, VEB Verlag Technik, Berlin 1988, 218 S., DDR 19,50 M

Der VEB Verlag Technik brachte ein von einem Autorenkollektiv verfaßtes, recht ansprechend aufgemachtes Lexikon der Computertechnik (1 000 Stichwörter) auf den Markt, dessen Erscheinen sehr zu begrüßen ist, wird doch damit eine weitere Lücke auf diesem Sektor zu schließen versucht. Denn es wird immer schwieriger, das komplexe Gebiet der Computertechnik und Datenverarbeitung mit seiner ständig größer werdenden Zahl neuer Fachbegriffe in seiner ganzen Breite zu übersehen.
Schon aus diesem Grunde wird es beinahe „Schicksal" eines jeden Lexikons auf diesem Fachgebiet sein, sich auf gewisse Teilbereiche bei den ausgewählten Stichwörtern beschränken zu müssen. Das ist auch in vorliegendem Band deutlich zu spüren.
Gleichwohl ist nicht zu übersehen, daß dieses Lexikon in vielerlei Hinsicht inhaltlich noch verbessert werden könnte.
Dazu drei Komplexbeispiele:
1. Aus dem gesamten Gebiet der Parallelrechentechnik wird lediglich der klassische Analogrechner als Stichwort berücksichtigt (hier sind die Definitionsgleichungen für Festwertgeber und Integrator – S. 15 – falsch angegeben). Diese Vorrangstellung bleibt unverständlich. Der Benutzer sollte sich doch in einem solchen Fachlexikon auch über Begriffe wie Arrayrechner, MIMD-, SIMD-Rechner, Transputer, Vektorrechner u. a. informieren können, die als Stichwörter aber fehlen.
2. Es ist sehr positiv zu vermerken, daß die Autoren daran gedacht haben, an gegebener Stelle auch historische Hinweise (in Form biographischer Kurzdaten) auf bedeutende Wissenschaftler zu geben, die für die Rechen- und Computertechnik Wesentliches geleistet haben. Denn dadurch wird der Bildungswert eines Lexikons zweifellos erhöht und ein noch größerer Nutzerkreis angesprochen. Die aufgeführte Ahnenreihe der Computerpioniere (Babbage, Aiken, Hollerith, v. Neumann, Zuse) könnte aber durchaus noch um die Namen W. Schickard (Erbauer der ersten mechanischen Rechenmaschine), Lord Kelvin (Entdecker des Rückkoppelprinzips in der Analogrechentechnik), A. Turing (Schöpfer der „Turingmaschine"), L. Boltzmann („Boltzmannrechner"), N. Wiener (Begründer der Kybernetik), N. Wirth (Schöpfer von Algol, Pascal, Modula) ergänzt werden. Alle Zweierpotenzen 2^n werden für $n = -24$ bis $n = +24$ abgedruckt, aber Leibniz als Erfinder des Dualsystems und Erbauer der ersten Sechsspezies-Rechenmaschine wird nicht einmal genannt!
3. Das Schwergewicht wurde in vorliegendem Lexikon offenbar auf die Hardware gelegt. Es ist zu begrüßen, daß die Autoren dennoch wichtige Begriffe aus dem Software-Bereich mit aufgenommen haben (Basic, Logo, Lisp, Pascal; CP/M, Unix). Die Betriebssysteme MS-DOS, PC-DOS, UDOS werden zwar im Text erwähnt, fehlen aber bedauerlicherweise als Stichwörter. Außerdem vermißt man z. B. Angaben über Ada, LAN, Modula, Occam, Pearl, Prolog, Simula sowie über virtuelle Speicher bzw. virtuelle Maschinen.
Es bleibt zu wünschen, daß in einer evtl. zweiten Auflage zumindest einige der hier genannten Verbesserungshinweise Berücksichtigung finden können. *Dr. K. Biener*

Arithmetische Algorithmen der Mikrorechentechnik

von G. Jorke, B. Lampe u. N. Wengel, VEB Verlag Technik, Berlin 1989, 324 S., DDR 33,50 M, Best.-Nr. 5539165

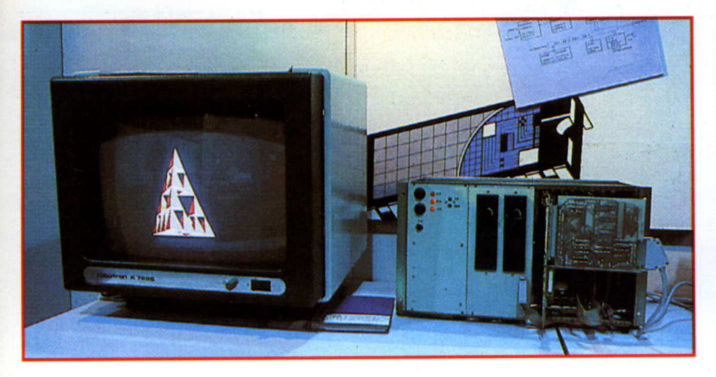
Der verstärkte Einsatz von Mikrorechnern für Echtzeitanwendungen in automatischen Steuerungen aller Arten erfordert die Nutzung effektiver Programmiertechnologien. Die Nutzung höherer Programmiersprachen wird hierbei in zunehmendem Maße im Vordergrund stehen, jedoch wird in vielen Fällen die maschinennahe Assemblerprogrammierung ebenfalls erforderlich sein. Höhere Verarbeitungsgeschwindigkeiten lassen sich mit Programmiertechniken erzielen, die die Mischung von aus Assembler- und Hochsprachen übersetzten Programmteilen gestatten. Hierbei spielt die Nutzung von Unterprogrammtechniken eine wesentliche Rolle. Das vorliegende Buch enthält eine Zusammenstellung wichtiger Unterprogramme für arithmetische und elementare geometrische Aufgaben; Konvertierung von Zahlendarstellungen, arithmetische Festpunkt- und Gleitpunktoperationen für Dualzahlen und Dezimalzahlen (Addition, Subtraktion, Multiplikation, Division, Quadratwurzel), Berechnung von Standardfunktionen (Taylorentwicklung, Polynomwertberechnungen, Exponential-, Logarithmus-, trigonometrische Funktionen). Im Kapitel über elementare geometrische Aufgaben werden Algorithmen und Beispielprogramme erläutert, die vor allem im CAD/CAM-Bereich die übersichtliche Darstellung und bequeme Manipulation großer Datenmengen oder komplizierte Vorgänge unterstützen und ermöglichen (Erzeugen von Bahnkurven für Werkzeugmaschinen und Roboter, Darstellung von Meß- und Kontrolldaten). Auch hier ist es zweckmäßig, zeitkritische Programmpassagen durch Assemblerprogramme auszutauschen. In diesem Abschnitt sind Verfahren und Programme angegeben, die für häufig wiederkehrende elementare Aufgaben erforderlich sind: Interpolation, Filterung oder Glättung, Koordinatentransformationen, Drehungen, Bahnkurven 2. Ordnung. Das Schlußkapitel stellt kurz zusammengefaßt wesentliche Aspekte der Formelabarbeitung in höheren Programmiersprachen dar. Es zeigt, wie auch für kompliziertere Formelausdrücke auf einfachem Weg Assemblerprogrammlösungen erreicht werden können.
Alle Darstellungen und Programmbeispiele orientieren sich an den in der DDR gängigen 8- und 16-Bit-Prozessoren U 881, U 880, U 8000, K 1810WM86 sowie Z8, Z80, Z8000, 8086 und 8087. Das Buch ist kurz und knapp abgefaßt, ohne daß Zusammenhänge verlorengehen. Es ist sehr übersichtlich gegliedert und hinterläßt in Aufmachung und Darstellung einen guten Eindruck. Zahlreiche instruktive Bilder unterstützen den Text. Vielfach werden die Programmbeispiele durch ausführlichere Grundlagen ergänzt. Dieses Buch ist vordergründig eine Darstellung für Programmierspezialisten, die diese Programmsammlung vielfältig als Nachschlagwerk nutzen können. Auch der weniger mit der Programmierung befaßte Leser wird aber eine Reihe von Anregungen finden; um so mehr, weil in kurzer, knapper Form nur das Wesentliche aufgeschrieben ist.

Prof. Dr. W. Fritzsch

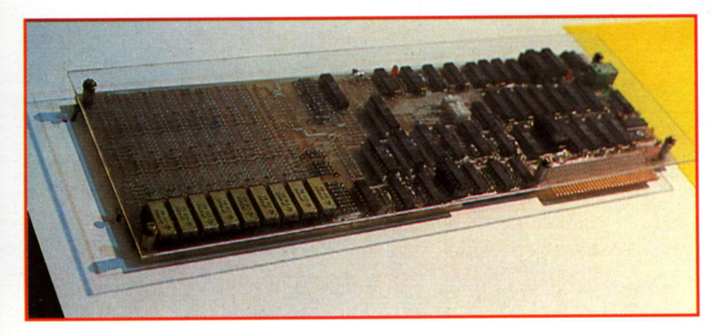
TERMINE

Softwarebörse zum Solitag

Am 26. November 1989 beteiligen sich am nun schon traditionellen Solibasar der Jugendmedien im Palast der Republik die Redaktionen JUGEND+TECHNIK mit Einchiprechner, practic mit Z 1013 und FUNKAMATEUR mit AC 1 und Mugler-PC. In der Zeit von 10.00 bis 17.00 Uhr werden Programme auf Kassetten und Disketten der Interessenten überspielt, Leiterplatten verkauft und EPROMs gebrannt. *R. Besser*



3 *Demo des SBC-WPU-80601 mit 800 × 800 Pixeln*



4 *4-MByte-RAM-Floppy für 16-Bit-PCs*



5. *Videoansteuerung VIS 3 für 8-Bit-PCs*



6 *Teilentwurfsarbeitsplatz für Gate-Arrays mit EC 1834*

Fortsetzung von der 2. US

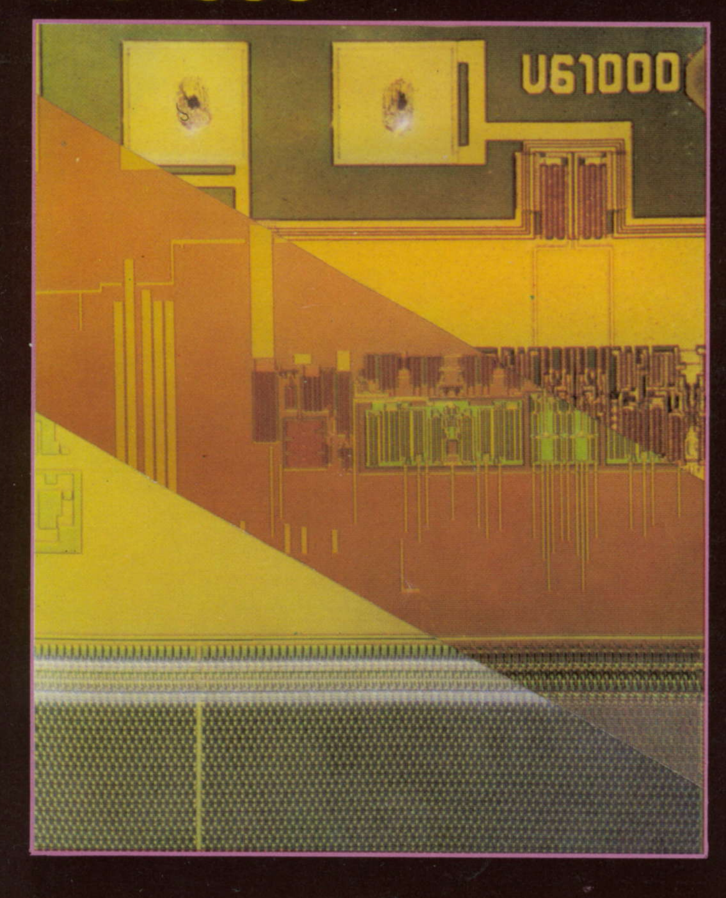
dem Farbmonitor veranschaulicht eine neue Qualität in Auflösung und Darstellung (Bild 3).
Das Zentrum für wissenschaftlichen Gerätebau der Akademie der Wissenschaften bot eine 4-MByte-RAM-Floppykarte und die Videoansteuerung VIS 3 an. Die **4-MByte-RAM-Floppy** (Bild 4) eignet sich für Speichererweiterungen des EC 1834 sowie von kompatiblen Rechnern nach EMS 3.2 (siehe auch MP 3/1989, Seite 89). Sie kann als schneller Datenspeicher in der Meßwerterfassung oder als Cache-Speicher für Harddiskzugriffe dienen. Mit dem obligatorischen Speichern von 9 Bit pro Byte wird die Paritätsprüfung gewährleistet. Eine Stützung bei externer Spannungszuführung sowie ein Autorefresh sind möglich. Die Karte wird mit 1-MBit-DRAMs U 61000 CC12 (Zugriffszeit 120 ns) bestückt; ausgestellt war die Bestückungsvariante mit 1 MByte. Die **VIS 3** (Bild 5, im Hintergrund) dient zur Ansteuerung von Farbmonitoren, die als vollgrafische Displays für die Bildverarbeitung und für Computergrafik eingesetzt werden können. Insbesondere ist sie für Einsatzfälle geeignet, wo ein bestehendes 8-Bit-Mikrorechnersystem durch ein grafikfähiges und farbtüchtiges Bildausgabesystem ergänzt werden soll. Die VIS 3 wurde bereits ausführlich in MP 3 und 11/1988 vorgestellt.
Zur Entlastung der Entwurfszentren und weil der Entwurf von anwendungsspezifischen Schaltkreisen (ASICs) in den Entwurfszentren nicht für jeden Nutzer effektiv ist, steigt die Nachfrage nach dem Entwurf mit dem PC. Der Teilentwurfsarbeitsplatz der Technischen Universität Karl-Marx-Stadt für den EC 1834 demonstrierte die Entwurfssystme **PC-GAD** (TU Karl-Marx-Stadt) und **MELGET** (Metallurgieelektronik Leipzig) für den Entwurf der Gate-Array-Systeme U 5200 und U 5300 (Bild 6). Das in Netzbeschreibungs- und Kommandosprache zum System ARCHIMEDES für den K 1840 kompatible PC-GAD erlaubt den Entwurf und die Simulation von Teilschaltungen mit bis zu 300 Makros. Eingehend können Sie sich darüber in unserer ASIC-Reihe in MP 3, 6 und 9/1989 informieren.
Das Schaltkreiszentrum in VEB Textimaelektronik Karl-Marx-Stadt hat in den drei Jahren seines Bestehens bis Anfang dieses Jahres Entwürfe für 30 Gate-Array-Schaltkreise U 5200/5300 und für 5 Standardzellenschaltkreise U 1500/1600 ausgeführt. Zum Symposium informierte das Schaltkreiszentrum über Logik-Cell-Arrays (LCAs), mit denen die Entwicklungszeiten von Leiterplatten mit ASICs stark verkürzt werden können. Bei geringen Stückzahlen können sie ASICs auch völlig ersetzen. LCAs werden – ähnlich wie EPROMs – am PC programmiert und wie ASICs eingesetzt. Damit erreichen die LCAs eine hohe Flexibilität. Sie sind änderungs- und testfreundlich; das Umprogrammieren ist in wenigen Minuten möglich. LCAs bestehen aus einer Menge von konfigurierbaren Logikblöcken (Flipflops und komplexe kombinatorische Logikfunktionen). Zur Kommunikation mit der Umwelt stehen Eingabe-/Ausgabeblöcke zur Verfügung. Alle Blöcke können uneingeschränkt miteinander verbunden werden. Ein Softwarepaket enthält Komponenten wie grafischer Editor, interaktives Berechnen der Signallaufzeiten, Makrozellenbibliotheken, automatisches Plazieren und Routen sowie eine dynamische Simulationsunterstützung.

Seit 1966 werden in Frankfurt(Oder) Bauelementesymposien durchgeführt. Sie stellen bis heute in der DDR in Größe und Bedeutung einzigartige applikative Veranstaltungen dar. Mit der Vielfalt der Informationsmöglichkeiten einschließlich des Verkaufs von Informationsmaterial und von Bauelementen wurde das Symposium seinem Ruf als wichtigste Informationsquelle für alle Anwender von integrierten Schaltkreisen in der DDR vollauf gerecht. MP-HK

Fotos: Hemke, Weiß

ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 10
S. 289–320

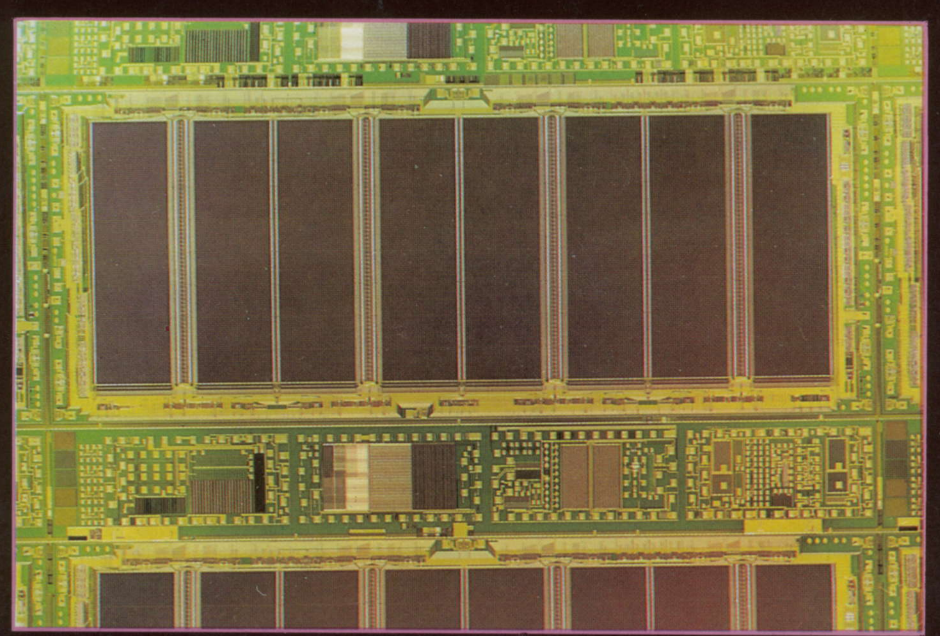
Der Megabitspeicher
U 61000



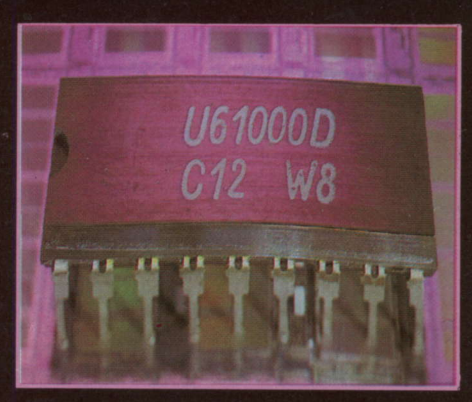
Das nebenstehende Bild zeigt einen Ausschnitt des Megabitspeichers U 61000 in den drei Bearbeitungsstufen (von unten nach oben):
* *nach der Strukturierung der Poly-Silizium-Speicherplatte*
* *nach der Strukturierung des Molybdänsilicids*
* *nach Fertigstellung und Zwischenprüfung.*

Auf dem Bild sind Teile der Speichermatrix und der Taktsteuerung sowie Bondinseln mit Testereindrücken zu erkennen.

Lesen Sie hierzu unseren Beitrag „Der Megabitspeicher U 61000" in diesem Heft.



Layout des Megabitspeichers U 61000 und des Testfeldstreifens für Technologieentwicklung und -kontrolle. Im Bild sind deutlich die vier Teilmatrizen der internen 256-K × 4-Organisation zu erkennen. Die gut leitende Alu-Wortleitung und die höherohmige Molybdänsilicid-Bitleitung führen zur langgestreckten Form der Teilmatrizen.



18poliges Duroplastgehäuse des U 61000 in Dual-in-line-Bauform (DIL)



MP

Heft 11 · 1989

Mikroprozessortechnik

VEB Verlag Technik Berlin

ISSN 0232-2892

TURBO-PASCAL

SYSTEM

Arithmetische Funktionen

Abs	liefert den absoluten Wert des Arguments
ArcTan	liefert den Arcustangens des Arguments
Cos	liefert den Cosinus des Arguments (Bogenmaß)
Exp	liefert das Ergebnis der Berechnung $e^{Argument}$, wobei e = 2.7182...
Frac	liefert die Nachkommastelle des Arguments
Int	liefert den ganzzahligen Anteil des Arguments
Ln	liefert den natürlichen Logarithmus des Arguments
Pi	liefert den Wert π (3.141592...)
Sin	liefert den Sinus des Arguments (Bogenmaß)
Sqr	liefert das Quadrat des Arguments
Sqrt	liefert die Quadratwurzel des Arguments

Routinen zur Stringverarbeitung

Concat	addiert eine anzugebende Folge von Strings
Copy	gibt einen Teilstring zurück
Delete	löscht einen Teil eines Strings
Insert	fügt einen String in einen anderen String ein
Length	liefert die augenblickliche Länge eines Strings
Pos	sucht eine Zeichenfolge innerhalb eines Strings und liefert ihre Position
Str	konvertiert einen numerischen Wert in einen String
Val	konvertiert einen String in einen numerischen Wert

Transfer-Funktionen

Chr	gibt von einer angegebenen Ordinalzahl das entsprechende ASCII-Zeichen zurück
Ord	gibt von einem angegebenen (ordinalen) Wert dessen Ordnungszahl zurück
Round	wandelt einen Realwert in einen LongInt und rundet dabei ab oder auf
Trunc	konvertiert einen Realwert in LongInt durch Abschneiden der Nachkommastellen

Ordinale Prozeduren und Funktionen

Dec	verringert den Wert einer Variablen
Inc	erhöht den Wert einer Variablen
Odd	liefert TRUE, wenn es sich bei dem Argument um eine gerade Zahl handelt, ansonsten FALSE
Pred	gibt den Vorgänger des Arguments zurück
Succ	gibt den Nachfolger des Arguments zurück

Routinen zur dynamischen Speicherverwaltung

Dispose	gibt den Speicherplatz einer dynamischen Variablen wieder frei
FreeMem	gibt einen dynamisch belegten Speicherbereich bestimmter Größe wieder frei
GetMem	belegt einen Speicherbereich der angegebenen Größe für eine dynamische Variable und setzt einen Zeiger auf den Anfang des Bereichs
Mark	hält den Zustand des Heaps in einer Zeigervariablen fest
MaxAvail	liefert die Größe des größten freien Heap-Bereichs zurück und ermittelt so, wieviele Bytes momentan maximal für eine dynamische Variable belegt werden können
MemAvail	liefert die Gesamtzahl der freien Bytes auf dem Heap
New	erzeugt eine dynamische Variable und setzt einen Zeiger auf den Anfang des durch diese Variable belegten Bereichs
Release	setzt den Heap auf den mit Mark festgehaltenen Zustand zurück

Zeiger- und Adreßfunktionen

Addr	liefert die Adresse des angegebenen Objekts
CSeg	liefert den momentanen Wert des CS-Registers
DSeg	liefert den momentanen Wert des DS-Registers
Ofs	liefert die Offset-Adresse (Adresse innerhalb eines Segments) des angegebenen Objekts
Ptr	konvertiert zwei Werte für Segment und Offset in einen Zeigerwert
Seg	liefert die Segment-Adresse des angegebenen Objekts
SPtr	liefert den momentanen Wert des SP-Registers
SSeg	liefert den momentanen Wert des SS-Registers

Verschiedene weitere Standardroutinen

Exit	beendet die Ausführung des momentanen Blocks und kehrt zum rufenden Block zurück
FillChar	füllt einen Speicherbereich angegebener Größe byteweise mit einem bestimmten Wert
Halt	sofortige Beendigung der Ausführung eines Programms und Rückkehr zur Betriebssystemebene mit wahlweiser Übergabe eines Returncodes
Hi	liefert das höherwertige Byte des Arguments
Lo	liefert das niederwertige Byte des Arguments
Move	kopiert eine Anzahl von Bytes von einem Speicherbereich in einen anderen
ParamCount	liefert die Anzahl der Kommandozeilen-Parameter, mit denen das Programm gestartet wurde
ParamStr	liefert einen Kommandozeilen-Parameter
Random	liefert eine Zufallszahl
Randomize	initialisiert den eingebauten Zufallszahlen-Generator
RunError	erzeugt einen Laufzeitfehler, der das Programm definiert abbricht
SizeOf	liefert die Größe des angegebenen Objekts in Bytes
Swap	vertauscht höherwertiges und niederwertiges Byte des Arguments miteinander und liefert das Ergebnis
UpCase	wandelt ein ASCII-Zeichen in sein großbuchstabiges Äquivalent um

Standardroutinen für alle Dateien

Assign	ordnet einer Datei-Variablen eine externe Datei über ihren Namen zu
ChDir	wechselt das momentan gesetzte Directory
Close	schließt eine offene Datei
Eof	prüft, ob das Ende der angegebenen Datei erreicht ist, und liefert einen Wahrheitswert zurück
Erase	löscht eine externe Datei
GetDir	ermittelt das momentan gesetzte Directory des angegebenen Laufwerks
IOResult	liefert den Status der zuletzt ausgeführten I/O-Operation als Integerwert zurück
MkDir	erzeugt ein Subdirectory
Rename	nennt eine Datei um
Reset	öffnet eine existierende Datei
Rewrite	erzeugt eine Datei und öffnet sie
Rmdir	löscht ein (leeres) Subdirectory

Standardroutinen für Textdateien

Append	öffnet eine existierende Datei für nachfolgende Schreiboperationen, neue Daten werden *angehängt*
Eoln	prüft, ob in der angegebenen Datei ein Zeilenende erreicht wurde, und liefert einen Wahrheitswert zurück
Flush	erzwingt das physische Schreiben eines Ausgabepuffers
Read	liest einen oder mehrere Werte aus der angegebenen Textdatei in die entsprechende Anzahl von Variablen
Readln	hat dieselbe Funktion wie Read, springt aber danach bis zum Anfang der nächsten Zeile in der Textdatei
SeekEof	prüft, ob das Ende der angegebenen Datei erreicht werden kann, und liefert einen Wahrheitswert
SeekEoln	prüft, ob in der angegebenen Datei ein Zeilenende erreicht werden kann, und liefert einen Wahrheitswert
SetTextBuf	ordnet einer Textdatei einen Ein-/Ausgabepuffer zu
Write	schreibt einen oder mehrere Werte in eine Textdatei
Writeln	hat dieselbe Funktion wie Write, schreibt aber danach automatisch ein Zeilenendezeichen

Standardroutinen für untypisierte Dateien

BlockRead	liest einen oder mehrere Records aus der angegebenen Datei in eine Variable
BlockWrite	schreibt einen oder mehrere Records aus einer Variablen in die angegebene Datei

Turbo-Pascal 5.0

UNITS

System	Laufzeitbibliothek von Turbo-Pascal, enthält den Standard-Sprachumfang und wird als einzige automatisch in jedes Programm aufgenommen.
Printer	unterstützt die Druckerausgaben
Crt	erlaubt die vollständige Kontrolle über Ein- und Ausgaben: direkte Zugriffe auf die Tastatur, Setzen der Videomodi, Farben, Fenster und Sound.
Dos	Schnittstelle zum direkten Aufruf von DOS-Funktionen (z. B. Datum und Uhrzeit, Suchen von Verzeichniseinträgen, Ausführen anderer Programme).
Overlay	Overlay-Manager, Routinepaket zur Verwaltung des von mehreren Overlays gemeinsam genutzten Speicherbereichs (Nachladen/Entfernen einzelner Units aus dem Speicher, Zugriffe auf .OVR-Dateien usw.)
Graph3	implementiert die Turtle-Grafik von Turbo-Pascal 3.0
Turbo3	stellt die Kompatibilität zur Turbo-Pascal Version 3.0 her
Graph	komplettes Grafik-Paket, unterstützt werden folgende Grafikkarten: CGA, EGA, VGA, Hercules, IBM 3270, MCGA und AT&T 6300

Standardroutinen für typisierte und untypisierte Dateien

FilePos	liefert die momentane Position des File-pointers innerhalb einer Datei zurück
FileSize	liefert die Größe einer anzugebenden Datei
Seek	setzt den Filepointer auf eine Position innerhalb der angegebenen Datei
Truncate	schneidet eine Datei an der momentanen Position ab

DOS

Interrupt-Prozeduren

GetIntVec	gibt die Adresse zurück, auf die ein Interrupt-Vektor zeigt
Intr	führt einen Software-Interrupt aus
MsDos	führt einen DOS-Funktionsaufruf (Int 21H) aus
SetIntVec	setzt einen Interrupt-Vektor auf eine bestimmte Adresse
SwapVectors	vertauscht die vom System belegten Interrupt-Vektoren mit den entsprechenden Variablen der Unit System

Prozeduren für Datum und Uhrzeit

GetDate	ermittelt das Kalenderdatum des Systems
GetFTime	ermittelt Datum und Uhrzeit der letzten Veränderung einer offenen Datei
GetTime	ermittelt die Uhrzeit des Systems
PackTime	konvertiert einen Record des Typs DateTime in eine Struktur von 4 Bytes (formal: Typ LongInt) für SetFTime
SetDate	setzt das Kalenderdatum des Systems
SetFTime	setzt Modifikations-Datum und Uhrzeit einer offenen Datei
SetTime	setzt die Uhrzeit des Systems
UnpackTime	konvertiert eine gepackte Struktur von 4 Bytes (formal: Typ LongInt) in einen Record des Typs DateTime. Derartige Strukturen werden von GetFTime, FindFirst und FindNext erzeugt

Statusfunktionen für Disketten- und Festplattenlaufwerke

DiskFree	liefert die Anzahl der freien Bytes auf dem angegebenen Laufwerk
DiskSize	liefert die Gesamtgröße des Datenträgers im angegebenen Laufwerk in Bytes

Prozeduren zur Bearbeitung von Dateieinträgen

FExpand	erweitert einen unvollständig angegebenen Dateinamen um den dazugehörigen Suchweg
FindFirst	sucht das angegebene bzw. momentan gesetzte Directory nach dem ersten Eintrag ab, der einem bestimmten Dateinamen und festgelegten Attributen entspricht
FindNext	setzt die mit FindFirst begonnene Suche fort
FSearch	sucht eine Liste von Verzeichnissen nach einem Dateieintrag ab
FSplit	zerlegt einen vollständigen Dateinamen in seine drei Komponenten Pfad, Dateiname und Dateinamenserweiterung
GetFAttr	ermittelt die Attribute einer Datei
SetFAttr	setzt die Attribute einer Datei

Sonstige Routinen

DosExitCode	gibt den Exit-Code eines mit Exec gestarteten Programms zurück
DosVersion	liefert die Versionsnummer von DOS
EnvCount	liefert die Anzahl der Einträge im Environment
EnvStr	liefert einen Eintrag aus dem Environment als String
Exec	startet ein Programm als child process, dem zusätzlich Kommandozeilen-Parameter übergeben werden können
GetCBreak	ermittelt, bei welchen Operationen DOS auf <Ctrl> <Break> prüft
GetEnv	liefert den zu einer Environment-Variablen gehörenden Eintrag als String
GetVerify	ermittelt den Stand des DOS-Flags Verify
Keep	beendet ein Programm und macht es speicherresident
SetCBreak	legt fest, bei welchen Operationen DOS auf <Ctrl> <Break> prüft
SetVerify	setzt das Verify-Flag von DOS

CRT

Routinen

AssignCrt	ordnet dem Bildschirm eine Textdatei-Variable zu
ClrEol	löscht sämtliche Zeichen ab der momentanen Position des Cursors bis zum rechten Fensterrand, ohne den Cursor zu bewegen
ClrScr	löscht den gesamten Fensterinhalt und setzt den Cursor in die obere linke Ecke des Fensters
Delay	hält die Programmausführung für die angegebene Anzahl von Millisekunden an (arbeitet unabhängig vom verwendeten Computermodell)
DelLine	löscht die Zeile, auf der sich der Cursor befindet und rollt die darunterliegenden Zeilen des Fensters um eine Zeile aufwärts. Die dadurch freiwerdende unterste Zeile des Fensters wird gelöscht.
GotoXY	setzt den Cursor auf die angegebene Spalten- und Zeilenposition. Die Zählung beginnt mit (1,1) und ist relativ zur oberen linken Ecke des Fensters
HighVideo	setzt die Zeichenfarbe auf *hervorgehoben*
InsLine	fügt eine Leerzeile an der Position des Cursors ein. Die folgenden Zeilen des Fensters werden entsprechend abwärts gerollt, die unterste Zeile verschwindet vom Bildschirm
KeyPressed	prüft die Tastatur und liefert TRUE zurück, wenn eine Taste gedrückt wurde, ansonsten ist das Ergebnis FALSE
LowVideo	setzt die Zeichenfarbe auf *halbe Intensität*
NormVideo	setzt die Zeichenfarbe auf *normal*
NoSound	schaltet den eingebauten Lautsprecher ab
ReadKey	liest ein Zeichen von der Tastatur, ohne Echo auf dem Bildschirm
RestoreCrt	setzt den Videomodus, der beim Start des Pogramms aktiv war
Sound	erzeugt einen Ton im eingebauten Lautsprecher
TextBackground	setzt die Hintergrundfarbe für folgende Textausgaben
TextColor	setzt die Zeichenfarbe für folgende Textausgaben
TextMode	setzt einen (Text-)Videomodus
WhereX	liefert die Momentane Spaltenposition (X) des Cursors relativ zur linken Fensterkante. Die Zählung beginnt mit 1.
WhereY	liefert die momentane Zeilenposition (Y) des Cursors relativ zur oberen Fensterkante. Die Zählung beginnt mit 1.
Window	definiert ein Textfenster auf dem Bildschirm

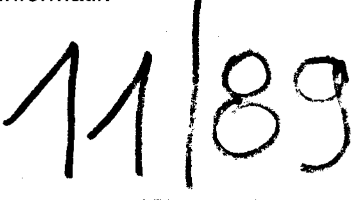
OVERLAY

Routinen

OvrInit	initialisiert die Overlay-Verwaltung und öffnet die .OVR-Datei des Programms
OvrInitEMS	kopiert die Overlay-Datei des Programms in den Speicherbereich der EMS-Karte
OvrSetBuf	legt die Größe des Overlay-Puffers explizit fest
OvrGetBuf	liefert die momentane Größe des Overlay-Puffers
OvrClearBuf	löscht den Overlay-Puffer

MP **Referenzkarte**

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232-2892





Mikroprozessortechnik, Heft 11 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR-1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 011 2228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titel Tatjana Stephanowitz

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

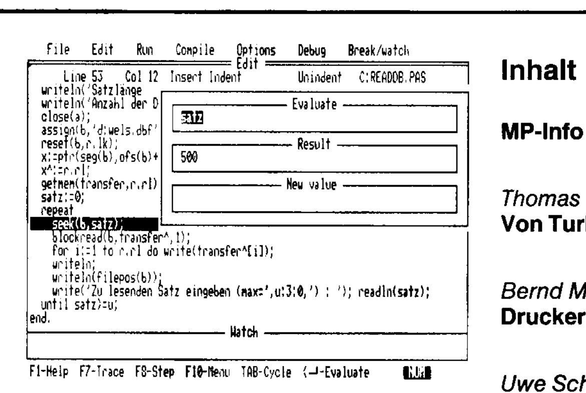
Redaktionsschluß 14. September 1989

AN (EDV) 49837

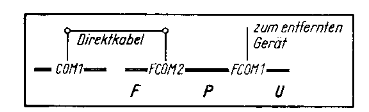
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

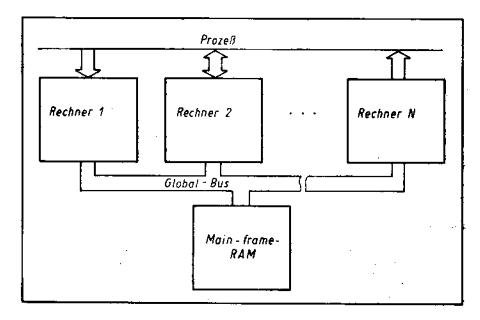
Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ústředni Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Ungarische VR:* P.K.H.I., Külföldi Elöfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR-7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR-7010 Leipzig



Turbo-Pascal, 1983 erstmals angeboten und seitdem ständig weiterentwickelt, erfreut sich zunehmender Beliebtheit und wird sicher auch in Zukunft zu den Standard-Programmiersprachen für 16-Bit-PCs gehören. Mit unserem Beitrag auf der **Seite 324** stellen wir Ihnen die gegenwärtig sehr weit verbreitete Version 5.0 vor. Die Referenzkarte auf der 2. und 3. Umschlagseite soll den Beitrag um ein praktisches Hilfsmittel ergänzen.



Unser Beitrag auf der **Seite 331** zeigt, wie Ihr 16-Bit-PC durch die Programmierung der Kommunikationskarte FPU um weitere acht V.24-Schnittstellen bereichert werden kann.



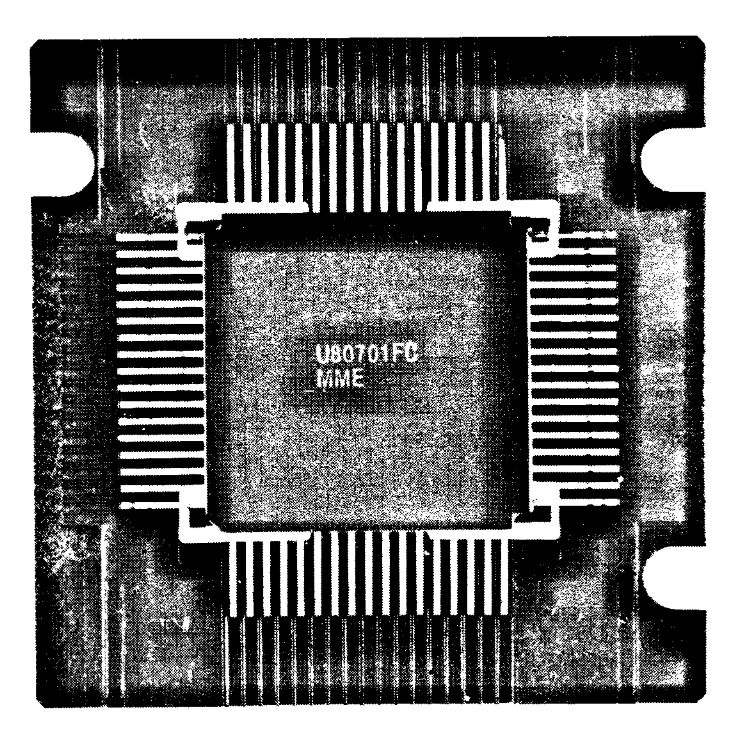
Der U 8000 bietet Unterstützung bei einem koordinierten Main-frame-RAM-Zugriff. Auf **Seite 333** wird ein flexibles und leistungsfähiges U 8000-Multiprozessorsystem mit geringem Hardwareaufwand vorgestellt.

Vorschau

Für das Heft 12/1989 bereiten wir für Sie unter anderem die folgenden Beiträge vor:
- Betriebssysteme der Echtzeitverarbeitung
- Programmieren in C
- Nutzerspezifische Gerätetreiber

Inhalt

Erster 32-Bit-Mikroprozessor aus Erfurt ermöglicht neue Rechnergeneration von Robotron



Neben einer breiten Sortimentsentwicklung kommt der Entwicklung von Spitzenprodukten als Prototypen neuer Technologien für die 90er Jahre große Bedeutung zu. Auf dem Gebiet der Festkörperschaltkreise wurde neben der Prototypenentwicklung dynamischer Speicher die Entwicklung leistungsfähiger Mikroprozessorsysteme als Einstieg in die Höchstintegration (VLSI-Technik) in der Direktive des XI. Parteitages der SED als zentrale staatliche Aufgabe festgelegt. Mit ihrer Erfüllung wird es der DDR möglich sein, den Bedarf höchstintegrierter Festkörperschaltkreise aus eigenem Aufkommen zu decken und damit Grundlagen für eine erforderliche Sortimentsbreite entsprechender strategischer Volkswirtschaftszweige, insbesondere der Rechentechnik, Steuerungselektronik und Informatik, zu schaffen.
Die ersten Arbeiten zur Entwicklung des ersten 32-Bit-Mikroprozessorsystems der DDR begannen im September 1986. Entsprechend den Erfordernissen der DDR wurde auf ein leistungsfähiges System zur Realisierung von Arbeitsplatzrechnern gehobener Leistungsklasse orientiert. Zu den bereits aus anderen Entwicklungsaufgaben resultierenden und verfügbaren Einzelkomponenten sollten 12 hoch- und höchstintegrierte Schaltkreise als einheitliches System entwickelt werden. Herzstück des Systems ist der 32-Bit-Mikroprozessor U 80701, der zirka 130000 Transistoren auf 85 mm^2 Chipfläche integriert. Dieser Mikroprozessor gestattet den Aufbau eines Arbeitsplatzrechners mit einer Leistungsfähigkeit von 1 Million Operationen pro Sekunde. Die Taktfrequenz beträgt 40 MHz. Der U 80701 verwaltet einen physischen Speicher von 16 MByte und einen virtuellen Speicher von 4 GByte. Er verfügt über eine integrierte Speicherverwaltungseinheit (MMU). Weiterhin verfügt er über einen seitenorientierten Schutzmechanismus (512 Byte je Seite) sowie eine Verwaltung der Zugriffsrechte (Protection). 175 Maschinenbefehle können genutzt werden. Alle Befehle sind orthogonal in ihrer Struktur und lassen für jeden der maximal 6 Operanden alle der 21 verschiedenen Adreßmodi zu. Für die nicht hardwaremäßig realisierten Befehle wird die Emulation auf Betriebssystemniveau unterstützt. Der Registersatz umfaßt 16 allgemeine Register mit 32 Bit Breite sowie 20 Prozessor- bzw. interne Register. Der Prozessor enthält einen Clockgenerator und einen Bulkspannungsgenerator.
Die 12 zum Gesamtsystem gehörenden Schaltkreise werden in den Kombinaten Mikroelektronik Erfurt und Carl Zeiss JENA produziert. Es wurde eine eigene Entwurfssoftware für hochkomplexe Logikschaltkreise, wie Mikroprozessoren, erarbeitet und am Beispiel des 32-Bit-Mikroprozessors U 80701 erfolgreich getestet.
Die mit dem 32-Bit-Mikroprozessorsystem U 80700 aufgebauten neuen 32-Bit-Rechner des VEB Kombinat Robotron, K 1820, sind mit dem in der DDR verfügbaren 32-Bit-Rechner K 1840 und mit international dominierenden Erzeugnissen der 32-Bit-Technik voll kompatibel. Die gegenwärtigen Hauptanwendungsgebiete des Rechners sind:
- Entwurf komplizierter Automatisierungssysteme
- Steuerung vollautomatischer Fabriken
- Informationssysteme zur Leitung der Volkswirtschaft
- Entwurf hochintegrierter Schaltkreise und komplizierter Mehrebenenleiterkarten
- Robotersteuerungen
- Telekommunikation, Satellitensteuerung.

Der K 1820 ist problemlos vernetzbar. An einen einzigen Rechner können 32 und mehr Bildschirmarbeitsplätze angeschlossen werden. An diesen lassen sich parallel unabhängige Arbeiten durchführen.
Durch die Unterstützung des Systems U 80700 wird eine virtuelle Speicherverwaltung ermöglicht, die in Verbindung mit international verbreiteten Betriebssystemen die Mehrprogramm- und Mehrnutzerarbeit erlaubt. Damit sind alle modernen Dialoganforderungen verschiedener Nutzer möglich, und Rechnerkapazität kann zur Bearbeitung von Hintergrundprozessen verwendet werden.
Durch den Verbund mehrerer Rechner kann durch eine aktiv geschaltete Redundanz eine Aufgabenumverteilung vorgenommen werden, so daß der Ausfall eines einzelnen Rechners praktisch ohne Auswirkung für den Nutzer bleibt.
Der K 1820 erfordert keine speziellen Aufstellbedingungen und ist aufgrund seiner geringen Größe an jedem Arbeitsplatz integrierbar.
Gegenüber dem bisher in der DDR breit angewendeten 8-Bit-Mikroprozessorsystem U 880 wird eine Materialeinsparung um den Faktor 10 erreicht.
Die in Ingenieurarbeitsstationen mit dem U 80700 mögliche hochauflösende Farbgrafik (Größenordnung 1 Million Bildpunkte, mehrere 100000 Farbnuancen) gestattet im Zusammenhang mit interaktiven Bedienmöglichkeiten den Einsatz für anspruchsvolle Konstruktions- und Darstellungsaufgaben.
Von besonderer Bedeutung sind die Möglichkeiten der Kommunikation von Ingenieurarbeitsstationen über leistungsfähige, das heißt auf hohe Geschwindigkeit orientierte Datenübertragungsverfahren. Mittels dieser Netzfähigkeit werden drei Grundprobleme gelöst: Erstens besteht eine effektive Basis zur Daten- und Informationsübertragung zwischen mehreren Rechnern (auch über große Entfernungen). Zweitens wird die Nutzung gemeinsamer Ressourcen – zum Beispiel teurer Peripheriegeräte wie Massenspeicher und elektrostatische Plotter – durch mehrere Ingenieurarbeitsstationen unterstützt. Außerdem besteht drittens die Möglichkeit, die dem Nutzer zur Verfügung stehende Rechenleistung durch Zusammenschaltung mehrerer Ingenieurarbeitsstationen zu erhöhen.
Mit dem System U 80700 und mit dem 1-MBit-DRAM U 61000 verfügt die Mikroelektronikindustrie der DDR nunmehr über alle Voraussetzungen, hochkomplizierte logische Systeme zu entwerfen und in die Fertigung umzusetzen. Dazu erforderliche Entwurfswerkzeuge wurden in Gemeinschaftsarbeit der Mikroelektronikkombinate, der Anwenderkombinate der Mikroelektronik, der Akademie der Wissenschaften und des Hoch- und Fachschulwesens der DDR erarbeitet.
Für den 32-Bit-Mikroprozessor wurde eine eigene Technologie, die zwei Metallisierungsebenen enthält, entwickelt. Die modernen Produktionsstätten des Kombinates Mikroelektronik am Standort Erfurt basieren auf einer in der DDR entwickelten Cleanroom-Konzeption und nutzen vorrangig technologische Spezialausrüstungen verschiedener Kombinate der DDR sowie Importe aus der UdSSR und den übrigen RGW-Ländern. Alle entscheidenden Materialien werden in der DDR hergestellt. Verwendet werden Siliziumscheiben aus dem VEB Spurenmetalle Freiberg, spezialreine Chemikalien für die Halbleitertechnik aus dem VEB Laborchemie Apolda, Keramikgehäuse aus dem Kombinat Keramische Werke Hermsdorf, Meßfassungen aus dem Kombinat Elektronische Bauelemente Teltow, Lacksysteme aus dem Kombinat ORWO sowie Lieferungen aus zahlreichen anderen Kombinaten. Der DDR ist es damit gelungen, sich beginnend beim Schaltkreisentwurf, der vorwiegend auf Rechentechnik des Kombinates Robotron aufbaut, über eigene Produktionskapazitäten in modernen clean rooms bis hin zur Versorgung mit wichtigen Materialien, eine umfassende eigene Basis zu schaffen. Darüber hinaus werden im Kombinat Mikroelektronik moderne Testsysteme zur Sicherung aller qualitativen Parameter entsprechend internationalen Maßstäben entwickelt und produziert.
Die in der DDR in den letzten Jahren erreichten Ergebnisse bei der Entwicklung anspruchsvoller Festkörperschaltkreise stellen eine entscheidende Grundlage für eine moderne Volkswirtschaft dar. Diese Grundlagen sichern im nächsten Jahrzehnt eine stabile Versorgung der Industrie mit den wichtigsten Sortimenten höchstintegrierter Schaltkreise aus Eigenaufkommen der DDR. *KME*

Textima-Schaltkreisentwürfe

40 anwenderspezifische Schaltkreisentwürfe für nahezu alle Bereiche der Volkswirtschaft hat das Kombinat Textima bisher bereitgestellt. Sie entstanden in einem eigens geschaffenen Schaltkreisentwurfszentrum des VEB Textimaelektronik Karl-Marx-Stadt, das 1986 als erste derartige Einrichtung im Maschinenbau der DDR die Arbeit aufgenommen hatte. Aus den Entwurfsunterlagen werden im Kombinat Carl Zeiss JENA nach Bedarf Schaltkreise in kleinsten Stückzahlen ebenso wie im Umfang von mehreren hunderttausend Stück produziert. Die in Karl-Marx-Stadt entworfenen Schaltkreise kommen in den Textima-Betrieben sowie in weiteren elf Industriezweigen zum Einsatz. Sie dienen insbesondere der schnelleren Verbindung des Maschinenbaues mit der Mikroelektronik, die ausschlaggebend für die Produktion von weltmarktfähigen Erzeugnissen ist. Die Entwicklung der mikroelektronischen Ausrüstungen erfolgt im engen Zusammenwirken mit wissenschaftlichen Einrichtungen, insbe-

sondere mit der Technischen Universität Karl-Marx-Stadt. Das Kombinat Textima beabsichtigt, die Kapazitäten für den Eigenbau von mikroelektronischen Baugruppen noch beträchtlich zu erweitern. Gegenwärtig werden Voraussetzungen geschaffen, daß ab 1990 kundenspezifische Schaltkreise bei Textimaelektronik selbst gefertigt werden können. Die konsequente Hinwendung zu Schlüsseltechnologien hat dazu geführt, daß schon in diesem Jahr im Kombinat Textima mehr als 70 Prozent aller Erzeugnisse über mikroelektronische Ausrüstungen verfügen. *ADN*

Erfurter Bezirks-MMM

Ergebnisse im Jugendobjekt Mikroelektronik, an dem FDJler von 15 Betrieben beteiligt sind, wurden auf der 32. Erfurter Bezirks-MMM im Juni vorgestellt. Gewichtige Leistungen haben beispielsweise junge Facharbeiter und Ingenieure aus dem VEB Mikroelektronik „Karl Marx" Erfurt aufzuweisen. Sie zeigten bei den 645 Exponaten unter anderem eine Erweiterungsleiterkarte, mit der sich die Rechenleistung von Sömmerdaer Computern auf das Drei- bis Vierfache steigern läßt. Das Kollektiv aus dem Forschungszentrum des Betriebes sammelte dabei Erfahrungen für den Einsatz des in Erfurt neuentwickelten schnellen 16-Bit-Mikroprozessors, der zum 40. Jahrestag der DDR in die Serienproduktion gehen sollte. Mit einem Sortiergerät für Siliziumscheiben war das MMM-Kollektiv „Franz Jakob" vertreten. Die Neuerung ermöglicht in der Halbleitervorfertigung durch Arbeitszeiteinsparungen einen ökonomischen Nutzen von 840000 Mark. *ADN*

Elektronische Bauelemente für Diskettenlaufwerke

Um eine zehnteilige Baureihe integrierter Schaltkreise für Diskettenlaufwerke ist das Produktionsprogramm im Betrieb Mikroelektronik des Kombinats Keramische Werke Hermsdorf erweitert worden. Die Hermsdorfer Widerstandsnetzwerke entsprechen den schaltungstechnischen und konstruktiven Forderungen des Laufwerkherstellers Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt. So beträgt die Einbauhöhe der flachen Netzwerke nur 8,5 Millimeter. Durch technologische Rationalisierung im betreffenden Bereich hatte das Hermsdorfer Kombinat Grundlagen für eine effektive Produktion in den benötigten hohen Stückzahlen geschaffen. *ADN*

Tage der Elektrotechnik und Elektronik der DDR in Bulgarien

„Tage der Elektrotechnik und Elektronik der DDR" gab es Mitte Juni in Sofia. Spezialisten führender Kombinate dieses Volkswirtschaftszweiges, darunter Elektro-Apparate-Werke, Carl Zeiss Jena, Mikroelektronik, Automatisierungsanlagenbau und Robotron, informierten über den erreichten wissenschaftlich-technischen Stand. Vorgestellt wurden unter anderem neue Halbleiterbauelemente, ein Leiterplattentester sowie verschiedene speicherprogrammierbare Steuerungen zur Automatisierung von Produktionsabschnitten. Eine Ausstellung demonstrierte die Zusammenarbeit zwischen Kombinaten der DDR und Wirtschaftsvereinigungen Bulgariens. An 16 Vorträgen von Spezialisten aus DDR-Kombinaten hatten rund 800 Techniker und Ingenieure aus bulgarischen Betrieben sowie von Wirtschaftsorganisationen teilgenommen. Derzeit bestehen vier Spezialisierungs- und Kooperationsabkommen innerhalb der elektronischen Meßtechnik, für die Produktion von Kühlschrankrelais, Elektroinstallationsmaterial und Großraumreinigungsgeräten. Gut entwickelt haben sich auch die Direktbeziehungen zwischen Betrieben beider Länder, die zum Teil schon traditionell sind, betonte der Stellvertreter des Ministers für Elektrotechnik und Elektronik der DDR, Rudi Wekker, zur Eröffnung, an der auch der Vorsitzende der Assoziation „Elektronika" Bulgariens, Iwan Tenew, sowie DDR-Botschafter Egon Rommel teilnahmen. Iwan Tenew würdigte die sich auch in der Elektrotechnik und Elektronik gut und dynamisch entwickelnde Zusammenarbeit zwischen der DDR und der VRB. Großen Anteil daran hätten die engen Arbeitskontakte auf allen Ebenen. Noch nicht alle Möglichkeiten würden genutzt; es gebe viele neue Gebiete, auf denen noch aktivere Zusammenarbeit beiderseits nützlich wäre. Die Tage hätten Anregungen dazu vermittelt. Geplant sei, 1990 diese Form des Erfahrungsaustausches mit der Vorstellung der bulgarischen Elektronik in der DDR fortzusetzen. *ADN*

Internationale Handelsmesse in Bulawayo

Mit einer Rekordbeteiligung öffnete die internationale Handelsmesse Simbabwes am 30. April in Bulawayo ihre Pforten. In der zweitgrößten Stadt des Landes boten 300 Aussteller aus 35 Staaten Afrikas, Amerikas, Asiens und Europas sowie 570 simbabwische Firmen bis zum 6. Mai ihre Exponate an. Erstmals auf der Messe vertreten waren Singapur, die Malediven, Mauritius und Sri Lanka.
Aus der DDR, die sich zum siebenten Mal an der Messe beteiligte, stellten auch die Außenhandelsunternehmen Elektrotechnik, Export-Import, Jenoptik, Robotron und Technocommerz aus. *ADN*

Internationaler Salon für Informatik in Paris

Der 40. Internationale Salon für Informatik, Datenübertragung und Bürotechnik (SICOB) fand Mitte April im Messezentrum Paris-Nord statt. Auf dem Salon, der zu den bedeutendsten Informatik-Fachmessen der Welt zählt, stellten mehr als 850 Firmen aus den führenden Ländern auf dem Gebiet der Mikroelektronik und ihrer Anwendung ihre Erzeugnisse vor. Traditionell gehörte das DDR-Kombinat Robotron zu den Ausstellern. Es bot vor allem elektronische Schreibmaschinen der Serie 3000, darunter erstmalig eine Maschine mit arabischen Schrifttypen, ferner Personalcomputer und periphere Geräte sowie elektronische Drucktechnik an.
Die Neuheiten des diesjährigen Salons betrafen vor allem Weiterentwicklungen bei der Anwendung der Mikroelektronik im Bürowesen, bei der Konstruktion und für Dienstleistungen. *ADN*

Jubiläumsmesse in Algier

An der 25. Algier-Messe Anfang Juni beteiligten sich rund 1000 Aussteller aus 33 Ländern und mehr als 200 staatliche und private Unternehmen des nordafrikanischen Landes. Die DDR, ununterbrochen auf der Exposition vertreten, beteiligte sich in diesem Jahr mit einer Kollektivausstellung. 17 Außenhandelsbetriebe und Consultingfirmen präsentierten auf 1200 Quadratmetern hochwertige, für die Entwicklung der algerischen Volkswirtschaft geeignete Industrieerzeugnisse, vor allem Maschinen und Anlagen sowie Produkte der elektrotechnischen und elektronischen Industrie.
Beim Besuch des DDR-Pavillons interessierte sich der algerische Ministerpräsident Kasdi Merbah besonders für die elektronische Schreibtechnik von Robotron, die ausschließlich mit arabischer Tastatur offeriert wurde. Ausführlich ließ er sich das Spitzenmodell „Erika 6007" erläutern, das erstmalig auf einer internationalen Messe ausgestellt wurde. *ADN*

Sowjetisch-bulgarisches Gemeinschaftsunternehmen

40000 Personalcomputer pro Jahr soll der sowjetisch-bulgarische Betrieb „Variant" für das Bildungswesen produzieren, der in Taschkent, der Hauptstadt Usbekistans, in Betrieb genommen wurde. Das Gemeinschaftsunternehmen entstand in einem Zeitraum von nur neun Monaten. Zu den Aktionären gehören die Ingenieurzentren „Wolna" und „Lidar" beim Komitee für Volksbildung der UdSSR und bei der Akademie der Wissenschaften der UdSSR sowie das bulgarische Kombinat für Mikroprozessortechnik in Pravez.
„Die den Arbeitskollektiven im Zuge der Umgestaltung der sowjetischen Wirtschaft eingeräumten neuen Rechte haben es ermöglicht, in kurzer Frist Betriebe zu schaffen, die den höchsten Anforderungen entsprechen", sagte der Vizepräsident der Akademie der Wissenschaften der UdSSR, Jewgeni Welichow, bei der Eröffnung des Unternehmens. Der Generaldirektor des Kombinates „Pravez", Plamen Watschkow, hob hervor, daß als Bedingung für die Gemeinschaftsproduktion niedrige Selbstkosten bei der Herstellung der Computer, die Entwicklung der Software und Service-Dienstleistungen vereinbart wurden. Es ist geplant, in Usbekistan mehrere Filialen von „Variant" einzurichten. Dadurch wird es bis 1991 möglich sein, alle Schulen in dieser Republik mit Computern auszustatten. *MP*

Kuba setzt Achtungszeichen bei Schlüsseltechnologien

Mikroelektronik „Made in Cuba" wurde noch vor wenigen Jahren mit Erstaunen zur Kenntnis genommen. Heute tragen Erzeugnisse des jüngsten Industriezweiges der Karibikinsel bereits internationale Messeauszeichnungen. Die UdSSR hat für dieses Jahr 10000 Bildschirmterminals in Kuba bestellt. Computertastaturen, auf die sich Kuba im RGW spezialisiert, werden in verschiedene Bruderländer exportiert.
Die Übernahme und Entwicklung von Schlüsseltechnologien wie der Mikroelektronik, Computertechnik oder Biotechnologie hat im Programm der KP Kubas besonderen Stellenwert bei der Vertiefung der sozialistischen ökonomischen Integration sowie der Erweiterung von Produktion, Forschung und Entwicklung. Ausgehend von der internationalen Dynamik dieser Bereiche besteht die Aufgabe, die materiell-technische Basis für Hochtechnologien zu konsolidieren und Ungleichgewichte in der Produktionsstruktur zu überwinden.
Schwerpunkt ist die Produktion von elektronischen Komponenten, Computern sowie industriellen Automatisierungsmitteln für den Export. Zugleich sollen einheimische Rohstoffe für die Elektronik gewonnen, medizintechnische Geräte, Kommunikationsmittel, Konsumgüter und Software verstärkt hergestellt werden.
Sicher kann und will sich Kuba nicht mit führenden Elektronik-Nationen messen, da das Land bei Schlüsseltechnologien erst am Anfang steht. Doch die jüngste Entwicklung läßt deutliche Fortschritte erkennen. So wurde im Dezember 1988 der erste einheimische Mikrochip vorgestellt. Und im Havannaer Zentrum für Software wird an CAD-Programmen gearbeitet. *ADN-Hempel*

Schlüsseltechnologien auf Dresdener Bezirks-MMM

Mit dem hochintegrierten Schaltkreis U 80617 gelang einem Jugendforscherkollektiv des VEB Forschungszentrum Mikroelektronik Dresden und des VEB Robotron-Elektronik Dresden die Entwicklung des ersten Schaltkreises einer neuen Klasse. Sie werden auf dem Gebiet der 32-Bit-Rechentechnik, die mit dem K 1840 vom Kombinat Robotron vor rund zwei Jahren begann, zum Einsatz kommen. Möglich wird damit eine bis zu 20mal höhere Arbeitsgeschwindigkeit bei verringertem Fertigungs- und Prüfaufwand. Die Entwickler stellten ihren Neuling im Konsultationsstützpunkt „FDJ und Schlüsseltechnologien" im August auf der 32. Messe der Meister von morgen des Bezirkes Dresden vor. *ADN*

Software ist vorhanden

Zu dem Beitrag „Entwicklung zuverlässiger Software in der Gerätetechnik" (MP 7/89, S. 212, Tafel 1) erhielten wir vom VEB Robotron-Rationalisierung Weimar den folgenden Hinweis:
Zur Steuerung der P 6000 bietet der VEB RRW folgendes Softwareangebot:
- Rahmensteuerprogramm „Systemsoftware für modulare Steuerung" (SPS 6000) mit prozeßorientiertem Steuerprogramminterpreter – SGM
- SGM-Editor und Übersetzer für SCP-Computer.

Von Turbo 3.0 zu Turbo 5.0

Professionelles für Profis

Thomas Bauer, Berlin

„Die Computerwelt erlebt derzeit einen Pascal-Boom, wie ihn selbst eingefleischte Anhänger dieser Programmiersprache nicht zu erhoffen wagten. Als Motor der Bewegung läßt sich unschwer ein preiswertes Produkt von Borland ausmachen, das auf allen Rechnern mit CP/M oder MS-DOS läuft: Turbo-Pascal." /1/
Mit diesem Zitat aus dem Jahre 1985 ist die Entwicklung von Turbo-Pascal treffend ausgedrückt. Pascal wurde Ende der 60er Jahre von N. Wirth entwickelt. Konzipiert war Pascal als Untermenge von ALGOL zum Zwecke der Lehre. Dem kommerziellen Einsatz standen die konzeptbedingten Einschränkungen, beispielsweise das Fehlen von Stringvariablen und von komfortablen Dateiverarbeitungsmöglichkeiten, entgegen. Turbo-Pascal mit seinen hervorragenden Eigenschaften bezüglich Schnelligkeit, Komfort und Entwicklungsumgebung brachte im Jahre 1983 die Wende.

Turbo-Pascal ist auf allen CP/M-Rechnern verfügbar und erreicht unter MS-DOS seine professionelle Ausprägung. Bei der Version 3.0, die einen Quasi-Industriestandard darstellt, liegen die Unterschiede zwischen der 8- und der 16-Bit-Version nur in den maschinennahen Sprachteilen. Die Version 3.0 ist für 8-Bit-Rechner gewissermaßen eingefroren – die Weiterentwicklung vollzieht sich konsequent zur Professionalität hin ausschließlich auf 16-Bit-PC und dort wiederum zu den schnellen Maschinen mit großen Speichern im MByte-Bereich hin. Turbo-Pascal Version 5.0 verkörpert als echte Ablösung der Zwischenversion 4.0 das heute bei der professionellen PC-Software erreichte Niveau, das sich in seinen Charakteristiken am ehesten mit früheren Sprachpaketen für Großrechner (Mainframes) vergleichen läßt. Komfort und Mächtigkeit lassen kaum noch Wünsche offen. Das Handbuch von Turbo 5.0 basiert auf dem der Version 4.0 zuzüglich eines Addendums von etwa 200 Seiten und ist nun mit insgesamt etwa 1000 Seiten gegenüber dem der Version 3.0 mit nur etwa 350 Seiten deutlich umfangreicher. Für Gelegenheitsprogrammierer ist die Version 5.0 inzwischen ein Gigant, bei dem die Frage nach der Verhältnismäßigkeit des Aufwandes zum Erlernen der Handhabung usw. zum erreichten Nutzen durchaus berechtigt erscheint. Zur Version 3.0 ist hier wenig zu vermerken. Es sei auf den MP-Kurs PASCAL verwiesen /2/. Interessanter sind die Unterschiede der nachfolgenden Versionen 4.0 und 5.0, weil sie den eigentlichen Fortschritt zum perfekten Softwarewerkzeug gebracht haben. Mit der Version 4.0 und vor allem 5.0 werden die Hersteller umfangreicher, kommerziell zu vermarktender Programmpakete angesprochen. Hier einige der wesentlichen Unterschiede von Turbo 4.0/5.0 zur Version 3.0:

- Statt COM-Dateien werden nun generell EXE-Dateien erzeugt und damit die 64-KByte-Grenze für Programmcode aufgehoben (Abkehr vom SMALL-MEMORY-MODEL).
- Modularität durch UNITs – getrennte Übersetzung und automatische Projektverwaltung
- zwei- bis dreimal schnellerer Compiler
- Kontextbezogene Hilfstexte und Bedienung mittels HOT-Keys (Tafel 1)
- noch komfortablere Benutzeroberfläche mit ausgefeilter Fenster- und Menütechnik
- *Intelligenter* Linker (Einbinden von OBJ-Dateien)
- DOS- und Grafikunterstützung im Standardsprachumfang
- Overlays wurden in der Version 4.0 ersatzlos gestrichen, sind aber in der Version 5.0 wieder enthalten.
- Erweiterter Sprachumfang mit neuen Standardprozeduren und neuen Datentypen
- Umstellungsunterstützung von Version 3.0 nach 4.0/5.0 (UPGRADE-Utility sowie UNITs TURBO3 und GRAPH3)
- Kommandozeilenversion des Compilers
- deutlicher Mehrbedarf an Speicherplatz (mindestens 384 KByte für Turbo 5.0).

Tafel 1 Einige wichtige HOT-Keys der Version 5.0

[F1]	aktiviert die HELP-Einrichtung
[F2]	Datei im Editor sichern
[F3]	Datei in den Editor laden
[F5]	aktuelles Fenster zoomen
[F6]	zwischen Edit- und Watch-Fenster umschalten
[F9]	MAKE-Prozedur starten
[F10]	schaltet zwischen Menüleiste und aktivem Fenster um
[Alt]–[F1]	letzten Hilfsbildschirm holen
[Alt]–[F3]	Datei aufgreifen via Pick-Liste
[Alt]–[F5]	schaltet zwischen Entwicklungsumgebung und MS-DOS-Bildschirm um
[Alt]–[F9]	kompiliert den im Editor befindlichen Quelltext
[Alt]–[X]	Turbo-Pascal verlassen
[Ctrl]–[F1]	Stichwort-Hilfe (nur im Editor verfügbar)
[Ctrl]–[F7]	Debugger: Eingabe eines Watch-Ausdrucks
[Ctrl]–[F9]	aktiviert Compile/Make und startet das Programm, wenn die Kompilation fehlerfrei verlaufen ist

Tafel 2 Numerische Datentypen der Version 5.0

Typ	Bytes	Bereich	
BYTE	1	0..255	
WORD	2	0..65535	
SHORTINT	1	−128..127	
INTEGER	2	−32768..32767	
LONGINT	4	−2147483648..2147483647	
REAL	6	−2.0E−39..1.7E38	
SINGLE	4	−1.5E−45..3.4E38	*
DOUBLE	8	−5.0E−324..1.7E308	*
EXTENDED	10	−1.9E−4951..1.1E4932	*
COMP	8	−9.2E18..9.2E18	*

Die mit * gekennzeichneten Datentypen sind nur mit einem mathematischen Koprozessor nutzbar.

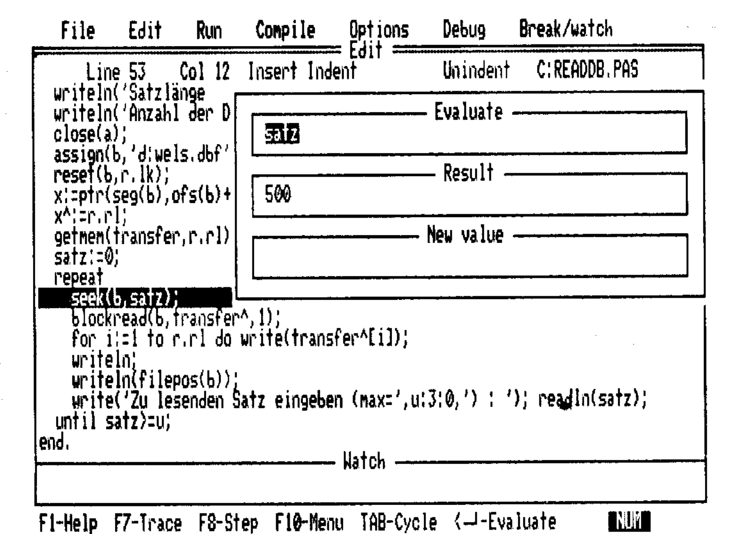
Die Version 5.0 kann sicher als ein in gewisser Weise abgeschlossenes Produkt für DOS-Rechner ähnlich der Version 3.0 für die CP/M-Rechner angesehen werden.
Was ist nun wesentlich an dieser Version? Sie enthält im *Professional Pack* sowohl einen Turbo-Assembler als auch einen Source-Code-Debugger. Damit wird nicht nur äußerlich ein in sich geschlossenes Paket präsentiert.

Professionelles Programmieren braucht Speicher; dem Limit von 640 KByte kommt man bei Nutzung des breiten Angebots an Softwaremöglichkeiten, die schon die Standardbibliothek TURBO.TPL bereithält, zuzüglich problemspezifischer kommerzieller Toolboxbibliotheken schnell nahe. Das mag ein Grund gewesen sein, die Overlays in die Version 5.0 nun doch wieder aufzunehmen. Ebenso wird mit der Unterstützung der EMS-Speichererweiterung Besitzern einer EMS-Karte (EMS 4.0 erweitert den PC-Speicher um 32 MByte, der in 4 × 16 KByte-Blöcken in den adressierbaren Bereich oberhalb der 640 KByte-Grenze ein- und ausgeblendet werden kann) die Möglichkeit geboten, diesen zusätzlichen Speicher sowohl mit der integrierten Entwicklungsumgebung als auch in eigenen Programmen zu nutzen. Das hilft 64 KByte des Hauptspeichers zu sparen und macht sich bei der integrierten Entwicklungsumgebung angenehm bemerkbar, wenn der Editor in einen 64 KByte-Bereich außerhalb des eigentlichen 640 KByte-Hauptspeichers verlagert werden kann. Ebenfalls positiv fällt die Möglichkeit zur Benutzung zweier Bildschirme nebst Adapter auf. Der aktive MS-DOS-Bildschirm wird für die Programmtestung genutzt, während Turbo-Pascal auf dem zweiten für das Betriebssystem normalerweise inaktiven Adapter läuft – eine vor allem bei der Entwicklung von Grafiksoftware zugegebenermaßen beeindruckende Arbeitsumgebung, die zur Anschaffung von zusätzlicher Hardware verleitet.
Eine weitere erwähnenswerte Neuerung bei der Version 4.0/5.0 ist die Installierbarkeit des Turbo-Pascal-Systems in jedes beliebige Verzeichnis und die Konfigurierbarkeit der Entwicklungsumgebung mittels einer Datei (Standardname TURBO.TP). Damit läßt sich die Turbo-Pascal-Software in jedes hierarchische Dateisystem auf der Festplatte problemlos einordnen. Der Aufruf des Turbo-Systems kann über den Pfad aus einem anderen Arbeitsverzeichnis erfolgen. Wenn sich nun in diesem Arbeitsverzeichnis eine Konfigurationsdatei befindet, so wird diese Datei gelesen. Die integrierte Entwicklungsumgebung wird dadurch mit den Einstellungen, die in der Konfigurationsdatei enthalten sind, für Compiler, Linker, Debugger, Verzeichnispfade usw. versehen. Damit kann prinzipiell zu jedem Projekt, das in einem eigenen Verzeichnis steht, eine entsprechende Konfigurationsdatei angelegt werden, die die integrierte Entwicklungsumgebung bei Arbeitsbeginn sofort mit den gewünschten Parametern versieht. An dieser Stelle sollen noch einmal die wesentlichsten Unterschiede der Version 5.0 zur Version 4.0 hervorgehoben werden:
- Unterstützung von Overlays
- EMS-Speichererweiterungs-Unterstützung (Expanded-Memory-Specification)
- verbesserter Linker
- 8087-Emulation
- berechnete Konstante
- integrierter Debugger
- prozedurale Parameter

Nun zu einigen Punkten im einzelnen.

Das UNIT-Konzept

Das UNIT-Konzept ist die interessanteste Neuerung gegenüber der Version 3.0. Damit wird die strukturierte Programmierung weiter unterstützt. Das Konzept erinnert sehr an Modula-2. Es bestehen damit nur noch wenige Gründe, das gegenüber dem Pascal-




```
File   Edit   Run   Compile   Options   Debug   Break/watch

  Line 27   Col 12   Insert Inde | Compiler      |   C:READDB.PAS
                                 | Linker        |
procedure TurboSeek(var x:file; j : | Environment |
begin seek(x,j) end;             | Directories   |

procedure seek(var x:f | Turbo    directory:  C:\SYS1\CL\TP5
var store : integer;   | EXE & TPU directory:
    v   : ^integer;    | Include directories:  C:\SYS1\CL\TP5\GRAFIK
    pos  : longint;    | Unit     directories:  C:\SYS1\CL\TP5\UNITS;C:\SYS1\C
begin                  | Object   directories:
  pos:=r.lk -------------- Object Directories ---------------
  v:=ptr(se
  store:=v^
  v^:=1;
  Turboseek(x,pos);
  v^:=store;
end;

begin
                              Watch

F1-Help  F5-Zoom  F6-Switch  F7-Trace  F8-Step  F9-Make  F10-Menu  NUM
```

Bild 1 Die integrierte Entwicklungsumgebung im Menüpunkt Debug/Evaluate

Bild 2 Das Optionsmenü Options/Directories/Object directories

Sprachkonzept modernere Modula-2 einem Turbo-Pascal in der Version 5.0 vorzuziehen. Eine UNIT ist eine Bibliothek von Deklarationen (Prozeduren, Funktionen, Typen, Variablen und Konstanten), die separat compiliert, bei Bedarf in ein Programm aufgenommen, und dort benutzt werden können. UNITs werden in Dateien mit dem Dateityp TPU gespeichert. Mit den UNITs werden zwei Vorteile eingebracht:

– Da jede UNIT bis zu 64 KByte groß sein darf, ist das Gesamtprogramm nur noch vom verfügbaren Gesamtspeicher begrenzt.

– Die Unterteilung des Programms in UNITs unterstützt die Entwicklung großer Programmsysteme sowie die Erstellung und Verwaltung von Standardbibliotheken. Damit wird eine arbeitsteilige Projektentwicklung gewährleistet.

Die Standard-Bibliothek TURBO.TPL enthält in der Version 5.0 die Standard-UNITs: SYSTEM, CRT, DOS, PRINTER und OVERLAY. Mit dem Programm TPUMOVER kann man die Standardbibliothek und andere UNIT-Bibliotheken verwalten. Das reservierte Statement

USES <name>[,<name>...];

dient zum Einbinden entsprechender UNITs in ein Programm. So kann man beispielsweise eine Toolbox, die in der Datei KERNEL.TPU als compilierte UNIT abgelegt ist, mit „USES KERNEL;" in ein beliebiges Programm aufnehmen. Allerdings müssen die UNITs der Version 4.0 zur Benutzung in der Version 5.0 neu compiliert werden. – Kein Problem, wenn die Quelltexte vorhanden sind; im anderen Falle eine fatale Situation für den Anwender, denn die Suche nach einem Umsetzungsprogramm für die UNITs ist vergeblich. – Ein solches gibt es nicht.

Tafel 3 Speicherbedarf der UNITs der Turbo-Pascal-Laufzeitbibliothek im DATA-Segment

Turbo-Pascal UNIT	Platzbedarf in Byte	
	Version 5.0	Version 4.0
SYSTEM	664	585
CRT	20	26
DOS	6	6
PRINTER	256	256
OVERLAY	10	–
GRAPH	1070	834
TURBO3	256	256
GRAPH3	0	0
Summe:	2282	1963

Fenster-Technik

Ein wesentlicher Unterschied zur Version 3.0 ist die Fenstertechnik mit den schon bei anderen Produkten der Turbo-Serie (z. B. Turbo-Basic, Turbo-C) zum Standard gehörenden Pull-Down-Menüs (Bild 1 und 2).

Während bei der Version 3.0 dem Editor der gesamte Bildschirm uneingeschränkt zur Verfügung gestellt wurde, wartet die Version 4.0 hier mit einer Neuheit auf. Der Bildschirm kann unterteilt werden in ein Output- und ein Edit-Fenster. Die Größe beider Fenster kann installiert werden. Das Output-Fenster enthält den DOS-Bildschirm, wie er sich vor dem Aufruf des Turbo-Systems präsentierte bzw. wie er nach dem Test eines Programms unter der integrierten Entwicklungsumgebung hinterlassen wurde. Im Edit-Fenster wird der Quelltext mit den gewohnten WordStar-kompatiblen Kommandos bearbeitet. Störend wirkt beim Editor der Version 5.0 allerdings, daß bei den Blockbefehlen die HOT-Keys [F7] (<Ctrl><KB>) und [F8] (<Ctrl><KK>) für die Blockmarkierung nun nicht mehr zur Verfügung stehen. Diese Tasten wurden für den Debugger verwendet, was einiger Gewöhnung bedarf. Zum effektiven Arbeiten ist das Zoomen sowohl des Edit- als auch des Output-Fensters auf volle Bildschirmgröße möglich. Als sehr nützlich erweist sich auch das *Drumherum* beim Quelltexteditieren; beispielsweise das *Picking* – ein schnelles Aufgreifen von Dateien – bei der Bearbeitung mehrerer eventuell zu einem Projekt gehörender Quelltexte. Durch Anwählen des Menüs *Option/Directories/Pick File* kann man eine sogenannte *Pickliste* anlegen, die auch als Datei gespeichert wird. In dieser Datei werden die Namen der letzten acht bearbeiteten Quelltexte abgelegt; sie stehen bei Arbeitsbeginn sofort zur Verfügung und können über die Picking-Funktion leicht gehandhabt werden.

In der Version 5.0 ist das *Output*-Fenster einem *Watch*-Fenster gewichen. In diesem Watch-Fenster können zur Laufzeit Variablenwerte verfolgt und auch manipuliert werden. Verantwortlich für diesen Service ist der integrierte Debugger, der eine wesentliche Erweiterung der Version 5.0 gegenüber 4.0 darstellt. Während der Fehlersuche stehen dem Programmierer über die Menüleiste und die HOT-Keys zahlreiche Überwachungsfunktionen zur Verfügung. Voraussetzung für die Arbeit mit dem Debugger ist die Beachtung notwendiger Compilerschalter-Voreinstellungen, die in der Mehrzahl der Fälle dann auch einen größeren Speicherbedarf erfordern (Symboltabellen etc.).

Integrierter Debugger

In der Version 5.0 hat sich nicht nur im Layout der Menüzeile etwas geändert. Einige Punkte sind neu hinzugekommen oder mit Erweiterungen versehen worden:

DEBUG
WATCH
RUN

Speziell bei *RUN* gibt es nun auch ein Pull-Down-Menü, das mehrere Menüpunkte enthält. Bis auf das ursprüngliche RUN, das die normale Programmabarbeitung startet, haben alle weiteren Subfunktionen im Zusammenhang mit dem integrierten Quellcode-Debugger ihre Bedeutung.

Mit diesem Debugger ist es beispielsweise möglich, die Werte von Variablen während der Laufzeit zu überprüfen – eine Aufgabe, deren Lösung in früheren Turbo-Pascal-Versionen nur durch explizite Ausgabeanweisungen im Quelltext herbeigeführt werden konnte. Bei der Version 5.0 wählt man mit Debug/Evaluate den entsprechenden Optionspunkt und kann nach Erscheinen eines dreigliedrigen Fensters die zu überprüfende Variable eintragen (Bild 1). Im mittleren der drei *Evaluate-Windows* erscheint der aktuelle Wert der betreffenden Variablen. Mit den Cursor-Tasten kann man in das dritte Fenster gelangen und dort sogar den aktuellen Wert der Variablen ändern. Steht der Cursor vor dem Aufruf von *Evaluate* auf einer Quelltextvariablen, so erscheint diese zur Kontrolle sofort im ersten Fenster. Wenn nun mehrere Variable eines Programms überwacht werden sollen, arbeitet man besser mit dem Watch-Fenster. Mittels HOT-Key [F6] kann man sich in das Watch-Fenster bewegen. Eine besonders effektive Möglichkeit bietet der Debugger im Trace- oder Einzelschrittmodus, womit in ganz hartnäckigen Fällen dem Fehler ähnlich wie beim Basic-Interpreter auf die Spur zu kommen ist. Im Trace-Modus können dabei Prozeduren übersprungen oder ebenfalls Schritt für Schritt untersucht

und beobachtet werden. Mit dem Setzen von Breakpoints gibt es noch eine weitere Option zur detaillierten Fehlerortung. Alles in allem stellt der Quellcode-Debugger ein sehr wirkungsvolles Instrumentarium zum gründlichen Austesten kompliziertester Algorithmen dar. Für solche Algorithmen lohnt sich die Anwendung des Debuggers und der Aufwand zum Erlernen seiner facettenreichen Anwendungsmöglichkeiten.

Erweiterter Sprachumfang

Der Sprachumfang wurde gegenüber der Version 3.0 drastisch erweitert. Es sind einige neue Datentypen hinzugekommen, die zum einen auch für Turbo-Pascal den Anschluß an den IEEE-Standard sichern sollen und zum anderen bei der Nutzung des Koprozessors üblich sind. Die numerischen Datentypen von Turbo-Pascal 5.0 zeigt Tafel 2.

Die frühere BCD-Arithmetik wird durch die IEEE-Datentypen abgedeckt (Verwendung von COMP bei Koprozessor, sonst LONGINT).

Pragmatisch eingestellte Programmierer werden einige Neuerungen in Richtung Aufweichung des strengen Typkonzepts von Pascal begrüßen:

– Untypisierte Zeiger mit dem Schlüsselwort POINTER. Ein derart definierter Zeiger zeigt nicht auf eine Variable eines bestimmten Typs, sondern ist mit allen Zeiger-Variablen zuweisungskompatibel.

– *Typecasting* wandelt jegliche Datentypen, seien es vor- oder selbstdefinierte, einfache oder strukturierte Typen, ineinander um. Als Einschränkung müssen beide Typen die gleiche Anzahl Bytes belegen. Das war in der Version 3.0 nur bei Aufzählungstypen direkt und bei anderen nur mit dem MOVE-Befehl möglich.

– Datentyp STRING ohne Längenangabe entspricht STRING[255]. Damit können solche Strings nun problemlos als Parameter für Unterprogramme benutzt werden.

Man sieht auch hier, die Professionalisierung bringt Ähnlichkeiten zu Großrechnersprachen wie PL/1 mit sich. Dazu gehören auch das UNIT-Konzept, die Overlay-Technik und vor allem der Linker und die Standard-Bibliotheken. Mit den von der Compilerversion und allen übrigen Softwarebestandteilen unabhängigen Dienstprogrammen MAKE, TOUCH und GREP können Programmkomplexe eines umfangreichen Projektes verwaltet werden. Auch diese Hilfsmittel sind vom Großrechnereinsatz her bekannt und für eine vorwiegend professionelle Softwareherstellung unentbehrlich. Insgesamt kann man sich jedoch des Eindrucks nicht erwehren, daß die Turbo-Pascal-Entwickler hier sozusagen ihr gesamtes eigenes Handwerkzeug *freigeben*.

Mit dem *Professional Pack* steht der labormäßigen Softwareentwicklung auf der Grundlage von Turbo-Pascal und Assembler ein mächtiges Entwicklungsinstrument zur Verfügung. Wer hier einsteigt, sollte für einige Jahre dabeibleiben, denn Einarbeitungsaufwand und Einleben in die Probleme dürften schon so groß sein, daß sich die Aufwendungen nur für etablierte Softwareentwickler über einen längeren Zeitraum mit entsprechenden Ergebnissen lohnen.

Für Maschinen mit 512 bzw. 640 KByte und dem ständig steigenden Angebot an hilfreichen residenten Programmen wird allerdings der Speicherplatz bald knapp werden, denn Turbo-Pascal 5.0 ist ein *Speicherfresser*, wenn man beispielsweise die Compilerdirektiven auf vollen Service einstellt.

Als Ausblick soll hier noch die Version 5.5 von Turbo-Pascal erwähnt werden. Mit dieser Version wird das Objektorientierte Programmieren als Option ermöglicht – bei vollständiger Kompatibilität zum alten Sprachumfang ohne diese Erweiterungen. Objekte ergänzen die reine *Container*-Funktion einer Variablen im traditionellen Sinne um deren Eigenschaften, Schnittstellen zu anderen Objekten und internen Funktionen. Damit wird mit nur vier neuen Schlüsselwörtern (object, virtual, constructor und destructor) vielen Pascal-Programmierern die Möglichkeit gegeben, sich in ihrer gewohnten Umgebung mit einer neuen, zukunftweisenden Programmiermethodik praktisch vertraut zu machen.

Generell ist bei der PC-Software eine Tendenz zur Professionalisierung mit einem Leistungsmaßstab analog zu etablierten Großrechnerinstallationen festzustellen. Bei der Leichtigkeit bezüglich Einsatz und Verfügbarkeit von Software wird oft übersehen: „Eines schickt sich nicht für alle!" Wer professionell programmiert, wird das Maximum an Möglichkeiten und die Installierbarkeit diverser Optionen freudig begrüßen – wer nur gelegentlich programmiert, und das dann notwendigerweise auf einem mittleren oder niedrigen Niveau, der wird eine ausgewogene Einfachheit (wie etwa die der Version 3.0) vorziehen.

Vielleicht muß man eine Untergliederung der Tätigkeit *Programmierer* akzeptieren, die zwischen *Werkzeugmachern* und *Werkzeugbenutzern* unterscheidet, wobei eben der Werkzeugbenutzer auch programmiert, aber seine Art zu programmieren liegt nicht mehr auf der Ebene von traditionellen Programmiersprachen, sondern er möchte komfortable Softwarewerkzeuge oder branchenspezifische Fachsprachen nutzen. Aus dieser Sicht ist die Professionalisierung moderner Programmiersprachen für PCs richtig und zweckmäßig. Dann aber mangelt es noch an branchenspezifischen, fachsprachlichen Subsystemen für die obengenannten Werkzeugbenutzer.

Die überwältigende Vielfalt an Möglichkeiten, die mit Turbo-Pascal im Professional Pack geboten wird, läßt keinerlei Gründe für eine Abkehr von Turbo-Pascal als industriemäßig erprobtes Softwareentwicklungswerkzeug erkennen. Im Gegenteil, unter Beachtung des Umfeldes an Toolboxroutinen usw. scheint es aus ökonomischen Gründen geradezu zwingend erforderlich, dieses Potential an Möglichkeiten zu nutzen. Der Umstieg auf andere Sprachen mag im konkreten Fall diesen oder jenen Beweggrund haben, der erzielte Gewinn wird dabei zweifelsohne mit dem einen oder anderen Verzicht erkauft werden müssen. Für die zur Zeit verfügbaren PCs unter MS-DOS oder OS/2 ist Turbo-Pascal 5.0 eine der mächtigsten Programmiersprachen.

Literatur

/1/ Kern, U.: CHIP-Test: Turbo-Pascal Version 3. CHIP, Würzburg (1985) 11, S. 118

/2/ Kofer, C.: MP-Kurs PASCAL. MP, Berlin (1987) 3, 11 und (1988) 3, 6, 9, 11

/3/ Heimsoeth Software GmbH & Co.: TURBO-PASCAL-Handbuch Version 3.xx (6. Auflage) Oktober 1986

/4/ Heimsoeth Software GmbH & Co.: TURBO-PASCAL-Handbuch Version 4.0. 1987

/5/ Heimsoeth Software GmbH & Co.: TURBO-PASCAL-Version 5.0 Addendum. 1988

Diskettenboxen

Von der Buchbinderei Gabriele Meichsner werden in verschiedenen Ausführungen die abgebildeten Diskettenboxen hergestellt.

Bild 1: ***Box für 50 Disketten; leinen- oder kunstlederbezogen***

Bild 2: ***Box für 1 Diskette; vorwiegend für Transportzwecke, mit Vorfallklappe, papierbezogen***

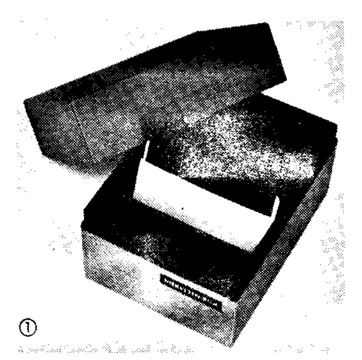
Bild 3: ***Fächerbox für 10 Minidisketten; leinen- oder kunstlederbezogen***

Bild 4: ***Box mit aufstellbarem Innenteil für 10 Disketten; leinen- oder kunstlederbezogen.***

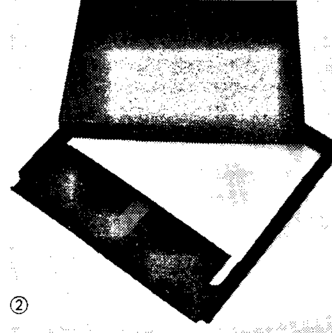
Schriftliche Bestellungen können an folgende Anschrift gerichtet werden:
Handwerkliche Buchbinderei,
Inh. Ramona Lubold,
Scheffelstraße 3–5, Plauen IV, 9900.

Anmerkung der Redaktion:

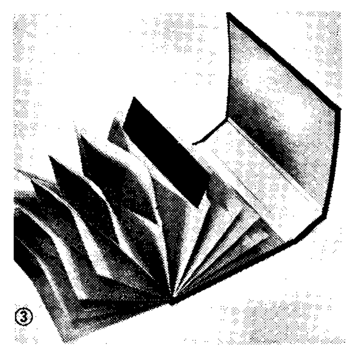
Die Disketten sollten zweckmäßigerweise nicht wie abgebildet, sondern so eingelegt werden, daß anstelle der Schreib-Lese-Öffnung das Etikett sichtbar ist.



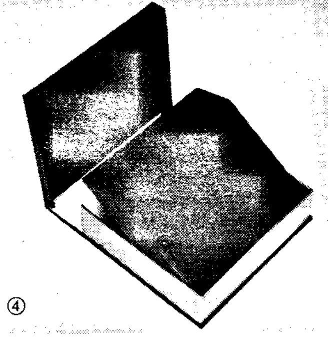
①



②



③



④

Druckerinitialisierung

Bernd Matzke, Delitzsch

Mit der umfassenden Einführung der Mikrorechentechnik nimmt die Typenvielfalt sowohl der Rechner als auch der Peripheriegeräte erheblich zu. Besonders nachteilig wirkt sich dies bei den Druckern aus, da gleiche Rechner oft mit verschiedenen Druckern gekoppelt sind und so ein Programm bei unterschiedlichen Druckern unter Umständen unterschiedliche Ergebnisse liefern kann. Außerdem werden einige Funktionen moderner Drucker (NLQ, unterschiedliche Zeichenhöhe usw.) durch Standardprogramme wie beispielsweise Textprozessoren nicht immer unterstützt.
Im Bild 1 möchte ich ein Drucker-Initialisierungsprogramm vorstellen, das durch die Verwendung der Programmiersprache Turbo-Pascal auf allen CP/M- und MS-DOS-Rechnern lauffähig ist.
Die Anpassung an den jeweiligen Drucker erfolgt durch Einbinden einer speziellen Datei, die den Vereinbarungsteil des Hauptprogramms entsprechend ergänzt. Zur Anpassung an andere Druckertypen ist lediglich diese INCLUDE-Datei zu ändern.

Aufbau der druckerspezifischen Datei

Ein Beispiel für eine solche Datei zeigt das Listing in Bild 2. Das Kernstück bildet das Array STEUER, in das alle gewünschten Druckerfunktionen aufgenommen werden, wobei mit jeder Funktion bis zu drei verschiedene Einstellungen des Druckers erreicht werden. Das Array besteht aus Records, deren Aufbau im folgenen erläutert wird.

① String *funktion*
Diese Zeichenkette beinhaltet den erläuternden Namen der mit den folgenden Steuerzeichen auszulösenden Funktion, beispielsweise *Zeichensatz* oder *Zeilenabstand*.

② Feld *anzahl*
Hier wird die Anzahl der möglichen Einstellungen eingegeben (z. B. 3 bei der Funktion Zeilenabstand, da der einzeilig, eineinhalbzeilig oder zweizeilig sein kann). Werte größer als 3 werden benutzt, um Funktionen mit einem von der Tastatur zu lesenden Parameter und die Art seiner Verarbeitung zu kennzeichnen.

③ Array *schrift*
In dieses Array werden die entsprechenden Einstellungen eingetragen, z. B. *ein* oder *aus*

④ Array *zeichen*
Die zum Drucker zu sendenden Steuerzeichen werden in dieses Array eingetragen. Das erste Byte enthält die Anzahl der zu sendenden Zeichen, die Steuerzeichen selbst stehen in den Bytes 2–4.

Ablauf des Programms

Nach der Anfangsinitilisierung werden die möglichen Funktionen auf dem Monitor angezeigt. Mit den Zifferntasten 2, 4, 6 und 8 kann die Auswahlmarkierung über den Bildschirm bewegt werden. Ich habe die Zifferntasten gewählt, weil deren Tastaturcode im Gegensatz zu dem der Kursortasten vom Betriebssystem und dem Tastaturtyp unabhängig ist.
In den Prozeduren *Markieren* und *Normal* wird der Name der durch die Variablen x und y gewählten Funktion entweder mit oder ohne eine entsprechende Auswahlmarkierung ausgegeben. Wenn der verwendete Rechner bzw. Monitor verschiedene Darstellungsarten (z. B. Inversdarstellung) erlaubt, so kann die Prozedur Markieren entsprechend abgeändert werden, was aber die universelle Anwendbarkeit des Programms einschränken kann.
Die Betätigung von <ET> aktiviert die jeweilige Funktion. Handelt es sich dabei um eine Funktion, die nur konstante Parameter an den Drucker senden soll, so wird die entsprechende Zeichenfolge ausgesendet und hinter dem Funktionsnamen die entsprechende Erläuterung angezeigt. Diese Erläuterung gibt an, in welchem Zustand sich der Drucker jetzt befindet, und nicht welche Einstellung beim erneuten Betätigen von <ET> erreicht wird. Wenn eine bestimmte Druckereinstellung mit verschiedenen Funktionen erreicht werden kann (z. B. Einstellung des Zeilenabstandes im Beispiel), so ist natürlich immer nur die zuletzt ausgewählte gültig.
Wird bei Funktionen im Array STEUER unter Anzahl eine 8 oder 9 eingetragen, erfolgt das Einlesen eines Parameters von der Tastatur. Gegebenenfalls wird er umgerechnet und mit den entsprechenden Steuerzeichen ausge-

```
PROGRAM druinst;
CONST offset  =   4; versch  =  16; dehnfac =  20;
      rechts  = '6'; links   = '4';
      hoch    = '8'; tief    = '2';
      CR      =  ^M;

{$I a:FX1000.DAT}

VAR x,y,                       {gerade angewaehltes Feld aus Array STEUER}
    xn,yn:INTEGER;
    taste:CHAR;
    merk:ARRAY[1..ymax,1..xmax] OF BYTE;
    ende:BOOLEAN;

PROCEDURE setzen ( verschiebung:INTEGER);        {Kursorpositionierung}
BEGIN
  gotoxy( Dehnfac * (x-1)+verschiebung, 2 * y + offset);
END;

FUNCTION parameter:BYTE;                    {Parameter von Tastatur einlesen}
VAR n:BYTE;
BEGIN
  setzen(versch);
  write('   ');                                  {alte Ausschrift loeschen}
  setzen(versch);
  buflen:=3;                               {Eingabe von maximal 3 Zeichen}
  read(n);
  parameter:=n;
  setzen(versch);
  write(n:3);                            {formatierte Ausgabe des Wertes}
  setzen(1);
END;

PROCEDURE send( steuerzeichen:Steuerzeichenfolge);
{Aussenden konstante Steuerzeichenfolge}
VAR n:BYTE;
BEGIN
  FOR n:=2 TO 1+steuerzeichen[1] DO write(lst,chr(steuerzeichen[n]));
END;

PROCEDURE sendx( steuerzeichen:Steuerzeichenfolge;b:BYTE);
{Aussenden Steuerzeichenfolge mit Parameter}
VAR n:BYTE;
BEGIN
  FOR n:=2 TO steuerzeichen[1] DO write( lst,chr(steuerzeichen[n]));
  write( lst,chr(b));
END;

PROCEDURE text( tex:s5);          {Ausschrift der erfolgten Einstellung}
BEGIN
  setzen(versch); write( tex);
END;

PROCEDURE output;          {Auswahl der zu sendenden Steuerzeichenfolge}
BEGIN
  WITH steuer[ y,x ] DO BEGIN
    CASE anzahl OF
      1,2,3:BEGIN                             {konstante Steuerzeichenfolge}
              IF merk[ y,x] < anzahl THEN merk[ y,x]:=merk[y,x] + 1
                                     ELSE merk[ y,x]:=1;
              text( schrift[merk[ y,x]]);
              send( zeichen[merk[ y,x]]);
            END;
{Steuerzeichenfolge mit zu berechnendem Parameter}
{835 ist die Laenge eines A4-Blattes in 1/72 Zoll}
      8    :sendx (zeichen[ merk[ y,x]],835 div parameter);
{Steuerzeichenfolge mit einfachem Parameter}
      9    :sendx (zeichen[ merk[ y,x]],parameter);
    END;
  END;
END;

PROCEDURE markieren;                      {Markieren der gewaehlten Funktion}
BEGIN
  setzen(1); write( '<'+steuer[y,x].funkt+'>');
END;

PROCEDURE normal; {Auschreiben eines Funktionsnamens ohne Markierung}
BEGIN
  setzen(1); write( ' '+steuer[y,x].funkt+' ');
END;

PROCEDURE anfang;                                 {Anfangsinitialisierung}
BEGIN
  clrscr;
  ende:=FALSE;
  writeln;writeln;
  writeln( 'Schalten Sie den Drucker ein und druecken Sie dann '+
           'eine beliebige Taste ');
  REPEAT UNTIL keypressed;
  clrscr;
  gotoxy( 10,1);
  write( ' Initialisierung',druckername);
  send( initial);                                      {Drucker ruecksetzen}
  FOR x:=1 TO xmax DO
    FOR y:=1 TO ymax DO BEGIN
      normal;                           {Alle Funktionsnamen ausschreiben}
      merk[y,x]:=1; END;             {jeweils ersten Parameter vorwaehlen}
  x:=1;  y:=1;  xn:=1;  yn:=1;        {Anfangswerte Auswahl 1. Funktion}
END;

BEGIN {druinst}
  anfang;
  markieren;                                     {erste Funktion markieren}
  REPEAT
    read( kbd,Taste);
    IF ((taste = rechts) AND (x < Xmax)) THEN xn:=x+1;
    IF ((taste = links ) AND (x > 1   )) THEN xn:=x-1;
    IF ((taste = hoch  ) AND (y > 1   )) THEN yn:=y-1;
    IF ((taste = tief  ) AND (y < Ymax)) THEN yn:=y+1;
    IF ((xn <> x) OR (yn <> y)) THEN BEGIN
      normal;                         {loeschen der alten Auswahlmarkierung}
      x:=xn;  y:=yn;
      markieren; END;              {Schreiben der neuen Auswahlmarkierung}
    IF (taste = CR) THEN                {aktivieren der Funktion mit <ET>}
      IF ((x=Xmax) AND (Y=Ymax)) THEN ende:=TRUE
                                 ELSE output;
  UNTIL ende;
  clrscr;
END.
```

Bild 1 Programm DRUINST.PAS

```
CONST xmax = 4; ymax = 7;
      druckername = ' FX 1000 ';

TYPE   steuerzeichenfolge  = ARRAY[1..4] OF BYTE;
       s5 = STRING[5];
       s13= STRING[13];
       sz = RECORD
              funkt   :s13;
              anzahl  :BYTE;
              schrift:ARRAY[1..3] OF s5;
              zeichen:ARRAY[1..3] OF steuerzeichenfolge;
            END;
       sza= ARRAY[1..ymax,1..xmax] OF sz;

CONST  initial: Steuerzeichenfolge = (2,27,64,0);   {Drucker - Reset}
       steuer : sza =(
(   ( funkt:'Pap.endef.';  Anzahl:2; schrift:('ein','aus','');
    zeichen:((2,27,57,0), (2,27,56,0), (0,0,0,0))),
    ( funkt:'';                Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';                Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';                Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0)))),

(   ( funkt:'Zeichenabst.';Anzahl:2; schrift:('ELITE','PICA ','');
    zeichen:((2,27,77,0), (2,27,80,0), (0,0,0,0))),
    ( funkt:'Schmaldruck'; anzahl:2; schrift:('ein','aus ','');
    zeichen:((1,15,0,0), (1,18,0,0), (0,0,0,0))),
    ( funkt:'Doppeldruck';anzahl:2; schrift:('ein','aus ','');
    zeichen:((2,27,71,0), (2,27,72,0), (0,0,0,0))),
    ( funkt:'Fettdruck';anzahl:2; schrift:('ein','aus ','');
    zeichen:((2,27,69,0), (2,27,70,0), (0,0,0,0)))),

(   ( funkt:'Prop.druck'; anzahl:2; schrift:('ein','aus ','');
    zeichen:((3,27,112,1), (3,27,112,0), (0,0,0,0))),
    ( funkt:'Kursiv.druck'; anzahl:2; schrift:('ein','aus ','');
    zeichen:((2,27,52,0), (2,27,53,0), (0,0,0,0))),
    ( funkt:'NLQ'; anzahl:2; schrift:('ein','aus ','');
    zeichen:((3,27,120,1), (3,27,120,0), (0,0,0,0))),
    ( funkt:'NLQ-Art'; anzahl:2; schrift:('ROMAN','SERIF','');
    zeichen:((3,27,107,0), (3,27,107,1), (0,0,0,0)))),

(   ( funkt:'Zeilenabst.'; anzahl:3; schrift:('1.0','1.5','2.0');
    zeichen:((3,27,65,12), (3,27,65,16), (3,27,65,22))),
    ( funkt:'1/72 Zoll';             Anzahl:9; schrift:('','','');
    zeichen:((3,27,65,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'je Seite';             Anzahl:8; schrift:('','','');
    zeichen:((3,27,65,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0)))),

(   ( funkt:'Halbes V'; anzahl:2; schrift:('ein','aus','');
    zeichen:((3,27,115,1), (3,27,115,0), (0,0,0,0))),
    ( funkt:'Unidirek.'; anzahl:2; schrift:('ein','aus','');
    zeichen:((3,27,85,1), (3,27,85,0), (0,0,0,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0)))),
(   ( funkt:'Linker Rand'; anzahl:9; schrift:('','','');
    zeichen:((3,27,108,0), (3,27,108,0), (3,27,108,0))),
    ( funkt:'Rechter Rand'; anzahl:9; schrift:('','','');
    zeichen:((3,27,81,0), (3,27,81,0), (3,27,81,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0)))),

(   ( funkt:'Zeichensatz'; anzahl:3; schrift:('ameri','dtsch','brit');
    zeichen:((3,27,82,0), (3,27,82,2), (3,27,82,4))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'';               Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0))),
    ( funkt:'Ende';                 Anzahl:0; schrift:('','','');
    zeichen:((0,0,0,0), (0,0,0,0), (0,0,0,0)))));
```

Bild 2 Include-Datei FX1000.DAT

sendet. Bei Bedarf können mit weiteren Kennzahlen individuelle Verarbeitungsmöglichkeiten für Parameter gekennzeichnet werden.
Beim Einlesen eines Parameters wird dessen Stellenzahl und die Länge des Eingabepuffers durch Zuweisung einer 3 zur vordefinierten Variablen BUFLEN begrenzt. Diese Zuweisung gilt nur für die unmittelbar folgende Read-Anweisung. Danach wird BUFLEN automatisch wieder auf den Maximalwert gesetzt.
Das Array STEUER sollte maximal 10 mal 4 Einträge beinhalten, da sonst die Darstellung auf dem Bildschirm gestört wird. Der letzte Eintrag muß für die Ausgabe der Zeichenkette *Ende* reserviert bleiben.
Weitere Einzelheiten sind dem kommentierten Listing zu entnehmen.

DOS-Gerätetreiber für SCOM-LAN

Uwe Schulze
Wilhelm-Pieck-Universität Rostock, Institut für Sozialistische Wirtschaftsführung

Uwe Schulze studierte von 1983 bis 1985 an der Technischen Universität Dresden, Sektion Informationsverarbeitung, und anschließend bis 1988 an der neu gegründeten Sektion Informatik der Wilhelm-Pieck-Universität Rostock. Zur Zeit ist er am Informatiklabor der WPU als Forschungsstudent tätig.

Moderne Betriebssysteme besitzen eine hardwareunabhängige Peripherieschnittstelle. Die Bedienung erfolgt über eigene Driver (Gerätesteuerprogramme). Im deutschen Sprachgebrauch hat sich der nicht sehr glückliche Begriff Treiber durchgesetzt. Gerätesteuerprogramme für die Standardperipherie gehören zum Lieferumfang von MS-DOS (im folgenden mit DOS bezeichnet). Für Zusatzgeräte liefert der Hersteller im allgemeinen entsprechende Treiber mit. Softwarepakete enthalten meist Steuerprogramme für verschiedene vorgesehene Hardware (z. B. Grafiktreiber). Mit diesem Konzept ist die Lauffähigkeit von Software auf recht unterschiedlicher Hardware garantiert; nur die Treiber werden ausgetauscht. Diese liegen als Datei (im allgemeinen mit der Erweiterung SYS) vor. Sie sind keine ausführbaren Programme, wie die vom Typ COM, EXE oder BAT – der Kommandointerpreter kann sie nicht starten. Die Einbindung in das Betriebssystem erfolgt über die Konfigurationsdatei CONFIG.SYS. Zusätzliche Treiber werden mit dem Befehl

DEVICE = <dateiname>

eingebunden. Informationen dazu finden Sie in /1/. Ein Entfernen von Gerätetreibern ist grundsätzlich nur durch Neustart des Systems möglich. Die DOS-internen Treiber NUL (*Schein*gerät), CON (Konsole), AUX (serielle Schnittstelle), PRN (Druckertreiber) sowie Treiber für Massenspeicher werden automatisch eingebunden. Sie bilden im Speicher eine Kette. Zusätzliche Treiber werden stets vorn eingebunden (aber hinter dem NUL-Gerät). So ist ein Überlagern möglich, da stets der erste Treiber eines spezifizierten Namens in der Kette angesprochen wird. Bekanntes Beispiel dafür ist der erweiterte Konsoltreiber ANSI.SYS, der unter dem Namen CON vor dem alten Konsoltreiber eingebunden und somit stets angesprochen wird. Er verarbeitet die erweiterten Escape-Sequenzen. Die Einbindung in das Betriebssystem ist in den Bildern 1 und 2 dargestellt.

Da die Schnittstelle für Gerätesteuerprogramme offengelegt ist, kann prinzipiell jeder selbst Treiber implementieren und damit neue Geräte einbinden oder die bisher benutzte Ansteuerung durch eine eigene ersetzen. Als Beispiele werden häufig verbesserte Konsol- oder Druckertreiber veröffentlicht. Da der Aufwand für die Implementation eigener Treiber aber recht hoch ist, empfehle ich Ihnen dafür einen anderen Weg, nämlich das Einhängen eigener Zusatzroutinen in die entsprechenden BIOS-Interrupts. Anders

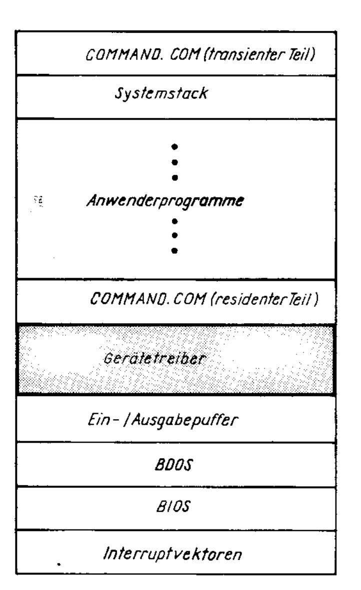


Bild 1 Speicheraufteilung von DOS

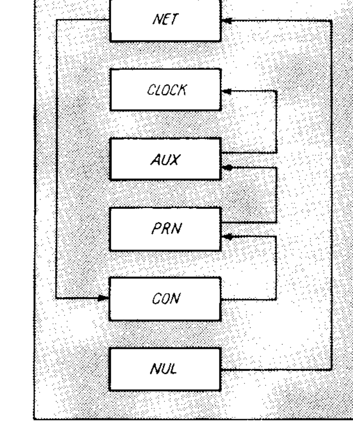


Bild 2 Die Treiberkette von DOS mit SCOM-Treiber

verhält es sich mit der Einbindung zusätzlicher Geräte in DOS. Hier empfehle ich die Implementation eines eigenen Treibers, um eine hardwareunabhängige Schnittstelle zu schaffen. Allerdings ist dazu eine Vorlage in Form eines Listings vorhandener Treiber vonnöten, da der vorgeschriebene Aufbau exakt eingehalten werden muß.

SCOM-LAN

Seit einigen Jahren existiert in der DDR das lokale Low-cost-Netz SCOM-LAN der Ingenieurhochschule Warnemünde /2/, das auf 8-Bit-Technik eine beachtliche Verbreitung erlangt hat. Durch die Nutzung als Front-end-Prozessor über die serielle Schnittstelle war auch die Nutzung unter DOS möglich – besonders interessant ist aber die seit kurzem verfügbare Slotkarte für den EC 1834 (vergleiche MP 8/1988, S. 253). Diese ist über Ports direkt ansprechbar. Die zur Zeit verfügbare Software programmiert diese Schnittstelle direkt oder greift auf vorgefertigte Routinen auf der Ebene der Transportschicht zurück. Anwendungen (vom einfachen Filetransfer bis zu verteilten Applikationen) haben eine komfortablere Schnittstelle verdient. Warum soll ein Filetransfer nicht mit dem DOS-Befehl

COPY <dateiname> NET

oder

COPY NET <dateiname>

abgewickelt werden können? Oder warum soll man auf eine komfortable Übertragung in Turbo-Pascal mit den Befehlen

write (NET, record)

bzw.

read (NET, record)

verzichten? Bekanntlich erfolgt unter DOS eine völlige Gleichbehandlung von Dateien und logischen Geräten (wie von UNIX bekannt). Diese Eigenschaft macht man sich beispielsweise bei der Nutzung des DOS-Befehls

COPY CON <dateiname>

als Mini-Editor zunutze.

Implementation

Das Betriebssystem DOS unterscheidet grundsätzlich zwei Arten von Treibern:
- Zeichentreiber und
- Blocktreiber.

Erstere bedienen vorrangig zeichenorientierte Geräte: Tastatur, Bildschirm, Drucker und serielle Schnittstellen. Blocktreiber sind den Massenspeichern vorbehalten; sie behandeln deren typische Aufgaben: Berechnung von Spur- und Sektornummern, logische Blockung, Mediumwechsel. Ein Netztreiber ist vom Wesen her zeichenorientiert, obwohl er paketorientiert arbeitet (was einer Blockung entspricht). Er weist aber keine der oben genannten Eigenschaften von Blocktreibern auf. Hingegen arbeitet auch ein Druckertreiber stets nur von Pufferfüllung zu Pufferfüllung.

Alle Treiber besitzen einen festen Aufbau, der mit Akribie eingehalten werden muß. Sie beginnen auf der relativen Adresse 0 (ORG 0 oder keine ORG-Anweisung) und müssen mit dem Dienstprogramm EXE2BIN umgewandelt werden. Es gelten damit die gleichen Restriktionen wie für COM-Dateien: keine Verwendung von absoluten Adressen. Auf Adresse 0 muß sich der Device-Header befinden (Bild 3). Er spezifiziert neben dem Attributwort (das die Eigenschaften des Treibers festlegt) und dem Namen die Adressen zweier Eintrittspunkte: der Strategie- und der Interrupt-Routine. Die Strategie-Routine (Bild 4) speichert lediglich die Adresse des in ES:BX übergebenen Parameterblocks (Request-Header – Bild 5) und gibt die Steuerung sofort ans DOS zurück. Anschließend ruft das Betriebssystem die Interrupt-Routine, die die eigentliche Ausführung vornimmt. Dieses scheinbar umständliche Vorgehen ist künftiger Multitasking-Arbeit geschuldet, bei der die Anforderungen in eine Warteschlange eingeordnet und unabhängig davon von einem Scheduler abgearbeitet werden.

Funktionalität

Ein Treiber kann 17 Funktionen unterstützen (Funktionscode 0 bis 16), wobei die Funktionen 13–16 erst ab DOS 3.0 vorgesehen sind. Beim Aufruf wird die Funktionsnummer im Request-Header (Bild 5) übergeben. Die Interrupt-Routine prüft auf Zulässigkeit und verzweigt zur entsprechenden Unterfunktion. Bild 6 zeigt den Sprungvektor und die Interrupt-Routine des SCOM-NET-Treibers.

Funktion 0: Die Initialisierungsroutine erfüllt folgende Aufgaben:

① Initialisieren der jeweiligen Schnittstelle
② Installationsmeldung für den Nutzer
③ Rückgabe des ersten freien Bytes nach dem Treiber
(Request-Header + 14).

Damit obliegt es dem Programmierer, den Initialisierungsteil, der nur bei der Installation aufgerufen wird, anschließend wieder freizugeben, sofern er am Ende des Treibers liegt. Dasselbe gilt für Meldungen und Konstantenfelder, die eventuell für die Initialisierung benötigt werden. Der SCOM-Treiber zeigt, daß diese Maßnahme durchaus sinnvoll ist, denn die Konvertierung der Adreßparameter, der Schnittstelleninitialisierung und das Setzen der eigenen Geräteadresse belegen mehr als 100 Zeilen im Quelltext.

Für die residente Installation des Treibers ist das DOS verantwortlich. Auf keinen Fall darf mit den DOS-Unterfunktionen 31H (Exit and Remain Resident) oder 48H (Allocate Memory) Speicher für den Treiber reserviert werden! Der SCOM-Treiber ruft an dieser Stelle aus der SCOM-Transportschicht die Funktionen
- Initialisierung und
- Setzen der eigenen Geräteadresse

auf /2/.

Funktion 1: Diese Routine gilt nur für Blocktreiber. Das DOS prüft, ob das Medium gewechselt wurde. Der SCOM-Treiber setzt nur das Fertig-Bit im Statusfeld (siehe Bild 7).

Funktion 2: Diese Routine gilt nur für Blocktreiber. Der BIOS-Parameterblock wird erzeugt.

Funktionen 3 und 4: Beide Funktionen unterscheiden sich nur dadurch, daß die Funk-

```
;**************************************
;*         Device Header              *
;**************************************

        dw      -1,-1               ; next driver
        dw      1100000000000000b   ; 1-char driver
                                    ; 1-IOCTL
                                    ; 0-with busy-status
                                    ; 0
                                    ; 0 extended DOS 3.0 fct.
                                    ; 0000000 - reserved
                                    ; 0 no NUL device
                                    ; 0 no CLOCK device
                                    ; 0 no default Input
                                    ; 0 no default Output
        dw      offset STRAT
        dw      offset INTR
        db      "NET     "
```

Bild 3 Allgemeiner Aufbau des Device-Headers sowie Device-Header des SCOM-Treibers

Offset	*Länge*	*Feld*
00H	*2 Byte*	*Offsetadresse des nächsten Treibers*
02H	*2 Byte*	*Segmentadresse des nächsten Treibers*
04H	*2 Byte*	*Geräteattribut*
06H	*2 Byte*	*Offsetadresse der Strategie-Routine*
08H	*2 Byte*	*Offsetadresse der Interrupt-Routine*
0AH	*8 Byte*	*Name des Treibers*

Bild 4 Strategie-Routine des SCOM-Treibers

```
;**************************************
;*       Strategy - Routine           *
;**************************************

STRAT    proc    far

         mov     word ptr cs:HeadPtr,bx ; Request-Header pointer
         mov     word ptr cs:HeadPtr+2,es
         ret

STRAT    endp

HeadPtr  dd      0
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR						BUSY	DONE	<			Error Code				>

```
ERR  = 1    :  Uebertragungs-Fehler (Fehlercode gesetzt)
BUSY = 1    :  Geraet ist aktiv
DONE = 1    :  Funktion ist ausgefuehert


Fehlercodes fuer Zeichentreiber

                 3 : unbekannte Funktion
                 9 : Papierende Drucker
                10 : Schreibfehler
                11 : Lesefehler
                12 : allgemeiner Geraetefehler
```

Bild 5 Request-Header und Statuswort

Offset	*Länge*	*Feld*
00H	*1 Byte*	*Länge des Request-Headers*
01H	*1 Byte*	*Nummer des Gerätes (nur Blocktreiber)*
02H	*1 Byte*	*Funktionsnummer*
03H	*2 Byte*	*Statuswort*
14H	*4 Byte*	*Adresse des Datenpuffers (bei Funktion 0 wird hier die Adresse des Treiberendes zurückgegeben)*
18H	*2 Byte*	*Anzahl zu übertragender Zeichen*

```
;*************************************
;*          Jump Table              *
;*************************************

CmdJmp:  dw      offset Init          ; 00 : Init
         dw      offset dummy         ; 01 : Media Check
         dw      offset dummy         ; 02 : Build BPB
         dw      offset Read          ; 03 : Read direct (IOCTL)
         dw      offset Read          ; 04 : Read
         dw      offset InpStat       ; 05 : Check Char in Buffer
         dw      offset InpStat       ; 06 : Check Input Status
         dw      offset dummy         ; 07 : Delete Input Buffer
         dw      offset Write         ; 08 : Write
         dw      offset Write         ; 09 : Write & Verify
         dw      offset dummy         ; 10 : Check Output Status
         dw      offset dummy         ; 11 : Delete Output Buffer
         dw      offset Write         ; 12 : Write direct (IOCTL)
         dw      offset dummy         ; 13 : Open Device
         dw      offset dummy         ; 14 : Close Device
         dw      offset dummy         ; 15 : Media Check
         dw      offset dummy         ; 16 : Write until busy

;*************************************
;*      Interrupt - Routine         *
;*************************************

INTR     proc    far
         push    ax
         push    bx
         push    cx
         push    dx
         push    si
         push    di
         push    bp
         push    ds
         push    es
         pushf
```

```
         mov     ax,cs
         mov     ds,ax

         mov     di,word ptr cs:HeadPtr  ; Request-Header pointer
         mov     es,word ptr cs:HeadPtr+2
         mov     bl,es:[di].cmd          ; command code
         cmp     bl,(jmpend-cmdjmp)/2    ; valid command code ?
         jle     cmd_ok                  ; yes, execute
         mov     ax,8003h                ; return code:
         jmp     short intr_end          ; unknown command

         ; command code ok : execute     -------------------
cmd_ok:  shl     bl,1                    ; mult. 2  (2 byte per adr.)
         xor     bh,bh                   ; clear bh
         call    cs:[bx+offset cmdjmp]     ; call command
         les     di,dword ptr HeadPtr    ; adr. --> es:di

         ; end of execution : return     -------------------
intr_end:
         or      ax,100h                 ; set ready bit
         mov     word ptr es:[di+stat],ax     ; store status in header

         popf
         pop     es
         pop     ds
         pop     bp
         pop     di
         pop     si
         pop     dx
         pop     cx
         pop     bx
         pop     ax

         ret

INTR     endp
```

Bild 6 Sprungverteiler und Interrupt-Routine des SCOM-Treibers

```
;*************************************
;*          Dummy                   *
;*************************************

dummy    proc    near

         xor     ax,ax                ; clear busy-bit
         ret

dummy    endp


;*************************************
;* Check Char (5) / Input Status (6) *
;*************************************

InpStat  proc    near

         mov     ax,0000001000000000  ; set busy-bit
         ret                          ; no char in buffer

InpStat  endp
```

Bild 7 Die Routinen Dummy und InpStat des SCOM-Treibers

```
program demo;

var scom_net:text;

begin

assign(scom_net,'NET');
rewrite(scom_net);
writeln(scom_net,'Alles Roger');
close(scom_net)

end.
```

Bild 8 Turbo-Pascal-Programm zur Demonstration der Nutzung des neuen DOS-Gerätes NET

tion 3 eine direkte Kommunikation zwischen Anwenderprogramm und Treiber über die DOS-Unterfunktion 44H (Input Output Control) ermöglicht. Dazu muß das IOCTL-Bit im Treiberattribut gesetzt sein. Funktion 4 wird durch die Lesefunktion des DOS aufgerufen. Der SCOM-Treiber führt die folgenden Funktionen der SCOM-Transportschicht aus:
- Eröffnen einer logischen Verbindung
- Senden eines Datenpaketes
- Schließen der Verbindung.

Dabei werden alle Parameter über eine Port-Schnittstelle an die Slotkarte geschickt, die über einen Prozessor Z 80 sowie über 64 KByte RAM verfügt. Sie enthält die Übertragungssoftware NIOS, die verbindungsorientiert arbeitet (Level 4 im OSI-Referenzmodell). Um sowohl eine gezielte Übertragung an einen bestimmten Empfänger als auch allgemeine Funktionen der Form

COPY <dateiname> NET

verwenden zu können, habe ich zu einem kleinen Trick gegriffen: Ist das erste Zeichen im Datenpuffer ein ESCAPE (1BH), so wird das folgende Byte als Zieladresse interpretiert. Bei eigenen Applikationen kann so eine Zieladresse angegeben werden. Bei Systembefehlen der oben genannten Form ist das erste Byte im Puffer ungleich ESCAPE, und es erfolgt ein Versenden an die Standardadresse 99.

Funktionen 5 und 6: Das DOS ruft diese Funktion auf, um zu prüfen, ob bereits ein weiteres Zeichen bereitsteht. Dies hat besonders für Tastaturtreiber Bedeutung. Bei Funktion 5 wird weiterhin das Zeichen bereitgestellt, ohne es aus dem Puffer zu entfernen. Damit kann das DOS jeweils ein Zeichen vorausschauen. Da der SCOM-Treiber paketorientiert arbeitet und eine Empfangsfunktion explizit aufgerufen werden muß, wird dieser Fall nie eintreten. Deshalb wird das Warten-Bit im Statusfeld stets auf 1 gesetzt (siehe Bild 7).

Funktionen 7 und 11: Durch diese Funktionen werden die Ein- bzw. Ausgabepuffer gelöscht. Damit kann beispielsweise verhindert werden, daß beim Abbrechen des Druckvorganges beim nächsten Druck der Restpuffer noch vorhanden ist. Da der SCOM-Treiber paketorientiert arbeitet, ist ein Leeren des Puffers nicht erforderlich.

Funktionen 8, 9 und 12: Dies sind Schreibfunktionen. Der SCOM-Treiber behandelt Funktion 9 (write and verify) genau wie Funktion 8, da ein Kontrollesen nicht möglich ist. Funktion 12 dient wieder der direkten Kommunikation Anwendungsprogramm – Treiber über IOCTL. Erfolgt eine explizite Adressierung der Empfangsstation, so befinden sich am Anfang des Puffers ein ESCAPE und die Absenderadresse.

Funktion 10: Das Betriebssystem prüft, ob der Schreibvorgang bereits abgeschlossen ist. Da die Steuerung von der Slotkarte stets erst an den Treiber zurückgegeben wird, wenn der Auftrag ausgeführt ist, kann das Warte-Bit immer auf 0 gesetzt werden.

Funktionen 13–16: Diese Funktionen werden erst ab DOS 3.0 unterstützt. Sie erlauben das Öffnen und das Schließen von Geräten, so daß zum Beispiel jeder Prozeß den Drucker anders initialisieren kann, ohne daß damit andere Prozesse beeinflußt werden. Interessant ist die Funktion 16 (wait until busy). Damit kann ein langsam arbeitendes Gerät seinen internen Puffer füllen, ohne daß alle Zeichen abgenommen werden und anschließend die Steuerung an das DOS zurückgegeben wird, während nebenbei gedruckt wird. Damit ist die Funktion für Spooler (Druckpuffer) geeignet. Für einen Netztreiber werden die Funktionen 13 bis 16 nicht benötigt.

Installation und Nutzung

Die Installation des Netztreibers erfolgt durch den Befehl

DEVICE = SCOM.SYS /n

in der Datei CONFIG.SYS, wobei n die eigene Geräteadresse (1–99) festlegt. Erfolgt keine Angabe, so wird als Standard 99 angenommen. Um die Verwendung gleicher Adressen auf verschiedenen Knoten zu vermeiden, sollte stets ein Parameter spezifiziert werden.

Ab jetzt kann das Netz wie eine Datei oder ein DOS-Gerät angesprochen werden. Bild 8 zeigt, wie ein String in Turbo-Pascal an das Netz verschickt werden kann. Es ist zu sehen, daß die Notation exakt dieselbe ist wie beim Schreiben in eine Textdatei. Ist der Treiber nicht geladen, so legt das Programm eine Datei mit dem Namen NET an und schreibt den String dort hinein – ein anschauliches Beispiel für die Gleichbehandlung von Dateien und logischen Geräten durch das DOS. Um die 8-Bit-Technik in eine allgemeine Nutzung einbeziehen zu können, stehen dort zwei kleine Programme zur Verfügung, die den COPY-Befehl vom DOS nachbilden und eine Datei absenden oder empfangen.

DOS ruft die Lese- und Schreibfunktionen von Zeichentreibern für jedes einzelne Zeichen auf. Damit würde im Falle des Netztrei-

bers für jedes Zeichen ein Paket verschickt werden, was die Geschwindigkeit entscheidend beeinflußt. Deshalb müssen die Zeichen in einem internen Puffer gesammelt und größere Pakete (256 Byte) verschickt werden. Dafür ist eine Dateiendekennung (Control Z) erforderlich. In Turbo-Pascal ist sie abschließend explizit zu schreiben; beim COPY-Befehl ist zum Absenden der Schalter /b und zum Empfang der Schalter /a zu verwenden.
Abschließend möchte ich noch anmerken, daß es trotz der gezeigten Vorzüge der Nutzung von DOS-Gerätetreiber mitunter günstiger sein kann, eine andere Schnittstelle zu nutzen. Als Beispiel seien verteilte Applikationen genannt, bei denen die Adressierung der Knoten im Netz eine große Rolle spielt und die bei der hier vorgestellten Lösung nur unzureichend unterstützt werden können.
Für die Spezifikation der SCOM-Transportschicht kann ich das SCOM-LAN-Handbuch /2/ empfehlen, und als Vorlage für eigene Treiberimplementationen sollten Sie /4/ verwenden.

Literatur

/1/ Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch. München; Franzis-Verlag GmbH 1987
/2/ Handbuch SCOM-LAN. Ingenieurhochschule Warnemünde 1987
/3/ IBM: Disk Operating System. Technical Reference 1984
/4/ Esders, E.: Treiberimplementation in MS-DOS. mc, München (1987) 12, S. 132

⊠ KONTAKT ☏

Wilhelm-Pieck-Universität Rostock, Institut für Sozialistische Wirtschaftsführung, Informatiklabor, Schröderstraße 23, Rostock, 2500; Tel. 37241, App. 30

Anwendung einer programmierbaren Kommunikationskarte

Christiane Ludwig, Roland Heckert
VEB Leitzentrum für Anwendungsforschung Berlin

In der DDR ist eine größere Anzahl von 16-Bit-Personalcomputern mit der programmierbaren Kommunikationskarte FPU (front processor unit) ausgerüstet. Erfahrungen zeigen, daß viele Anwender die Möglichkeiten dieser Karte nicht oder nur in sehr beschränktem Maße effektiv nutzen können. Aus diesem Grunde wird im folgenden Beitrag eine kurze Einführung in Aufbau und Funktion der FPU, die Nutzung der Programmschnittstellen für die Anwenderprogrammierung und eine Übersicht zu Software gegeben, die diese Karte unterstützt.

Aufbau und Funktion

Die FPU ist ein programmierbarer Mikrorechner mit einer Verarbeitungsbreite von 8 Bit. Sie besteht aus einem Mikroprozessor Z 80 mit einer Taktfrequenz von 6 MHz, einem Speicher von 8 KByte EPROM und 16 KByte RAM sowie aus zwei SIO- und zwei CTC-Bausteinen. Die Karte ist über einen direkten 62poligen Steckverbinder an den Systembus des PCs angeschlossen und bietet 4 asynchrone V.24-Schnittstellen.
Mit einem geeigneten Steuerprogramm kann die FPU für die Kommunikation des PCs über diese Schnittstellen geladen werden. Dadurch kann die FPU die gesamte Kommunikation über die V.24-Schnittstellen entsprechend der Auslegung der geladenen Steuersoftware völlig eigenständig abwickeln. Der PC wird lediglich mit der Übergabe und der Übernahme der Daten (einzeln oder geblockt) belastet. Dadurch kann ein PC neben den üblichen 2 V.24-Schnittstellen (COM1, COM2) noch weitere 8 V.24-Schnittstellen (2 FPU-Karten) (FCOM1 bis FCOM8) betreiben.
Die Hardwareadresse kann durch Steckbrükken auf der Karte zwischen 300H und 3FOH eingestellt werden. Die Sektoradresse für den Übergabe-/Übernahmebereich im PC ist ebenfalls durch Steckbrücken auf der Karte – abhängig von der konkreten Kartenversion – auf C000H, D000H oder E000H einstellbar.

Anwenderprogramme

Nach Installation der FPU-Firmware (siehe unten) stehen dem Anwender für die Programmentwicklung eine Erweiterung des BIOS-Interrupts 14 für den zeichenweisen Datenaustausch und ein BIOS-Interrupt 60 für den Blocktransfer zur Verfügung. Die Programmierung können Sie den Tafeln 1 und 2 entnehmen. Bei Neuentwicklung von Kommunikationssoftware für Rechner, die mit der FPU ausgerüstet sind (siehe unten), wurde auf die Benutzung des INT60 orientiert, um die Vorteile der FPU mittels geblockter Datenübergabe und -übernahme voll auszuschöpfen. Soll ein Programm jedoch sowohl mit der FPU als auch herkömmlichen Kommunikationskarten lauffähig sein, muß der INT14 benutzt werden. Bei der Programmie-

Tafel 1 INT14

	Eingabe			Ausgabe	
Funktion	AH	AL	DX	AH	AL
Initialisieren Schnittstelle	00	Initialisierungswerte (*)	Unit-	Leitungsstatus (**)	Modemstatus
Senden Zeichen	01	zu sendendes Byte	nummer COM1 – 0	Leitungsstatus	Modemstatus
Lesen Zeichen	02		COM2 – 1 FCOM1 – 2 FCOM2 – 3	0 Leitungsstatus	empfangenes Zeichen Modemstatus
Statusabfrage	03		FCOM3 – 4	Leitungsstatus	Modemstatus
Open	04		FCOM4 – 5	RC	(***)
Close	05		FCOM5 – 6	RC	
Cancel	06		FCOM6 – 7	RC	
Set redirection	07	Unitnummer Destination	FCOM7 – 8 FCOM8 – 9		
Status redirection	08				Unitnummer

(*) Initialisierungswerte:

Bitmaskierung in AL:

7	6	5		4	3	2	1	0	
Bitrate				Parität		Stopbits	Datenbits		
0	0	0	150	x	0 ohne	0 = 1	1	0	7
0	0	1	300	0	1 ungerade	1 = 2	1	1	8
0	1	0	600	1	1 gerade				
0	1	1	1200						
1	0	0	2400						
1	0	1	4800						
1	1	0	9600						
1	1	1	19200						

(**) Status:	Leitung	Modem
Bit 7	Time Out	Received Line Signal detect
Bit 6	Trans Shift Register Empty	Ring Indicator
Bit 5	Trans Holding Register Empty	Data Set Ready
Bit 4	Break detect	Clear to send
Bit 3	Framing Error	Data receive line signal detect
Bit 2	Parity Error	Trailing Edge Ring Detector
Bit 1	Overrun Error	Delta Data Set ready
Bit 0	Data ready	Data Clear to send

(***) RC-Rückkehrcodes:
0000 Funktion fehlerfrei ausgeführt
8000 Fehler über Funktion STATUS abfragen

Tafel 2 INT60

Funktion	Eingabe				Ausgabe	
	AH	ES:BX	CX	DX	AX	CX
Open	00			Unit-	RC	
Close	03			nummer	RC	
Read	01	Pufferadresse	max. einzulesende Bytes	FCOM1 – 0 FCOM2 – 1	RC	eingelesene Bytes
Write	02	Pufferadresse	zu übertragende Bytes	FCOM3 – 2 FCOM4 – 3	RC	übertragene Bytes
Readmode 1 2 3	04 05 06			FCOM5 – 4 FCOM6 – 5	Readmode	
Flush Input Output	07 08			FCOM7 – 6 FCOM8 – 7	RC	
Get Info Status Input Output	09 0A				RC	BX – Anzahl der Blöcke CX – Anzahl der Bytes

RC-Rückkehrcodes:
RFFF noch Zeichen im Puffer
R000 Funktion fehlerfrei ausgeführt
R0E0 Einheit bereits geöffnet
R0E1 Einheit nicht geöffnet
R0E2 Sendepuffer nicht leer (Funktion CLOSE)
R0E3 Einheit bereits geschlossen
R0E4 Empfangspuffer nicht leer (Funktion CLOSE)
R0E5 Überlauffehler
R0E6 Stopbit nicht erkannt
R0E7 Paritätsfehler
R0E8 Einheit nicht erreichbar
R0E9 externe Einheit nicht bereit
R0EA Sendepuffer voll
R0EB Befehl nicht gefunden
R0EC interner FPU-Fehler
R0ED Einheit nicht initialisiert
R – eingestellter Readmode oder 0

1 = readmode 1; Lesen aller Zeichen
2 = readmode 2; Lesen aller Zeichen zwischen STX (02) und ETX (03)
3 = readmode 3; Lesen aller Zeichen zwischen zwei CR (0D)

rung mit INT60 ist die jeweilige Schnittstelle (FCOM1 bis FCOM8) mit dem Programm FMODE (siehe unten) oder mit dem INT14-Programm zu initialisieren. Bei Programmierung mit INT14 kann die Schnittstelle durch FMODE oder direkt durch das Programm selbst initialisiert werden.

Software für die FPU unter DCP

Für das Betriebssystem DCP ist ein Grundsoftwarepaket verfügbar, das die Nutzung der FPU für die Entwicklung und die Anwendung von Kommunikationssoftware ermöglicht. Es besteht aus den Komponenten:
FPULOAD.EXE
– Installation der Interruptroutinen
– Eintragung der Adressen in die Interruptvektortabelle des DCP
– Laden und Starten der FPU-Firmware aus der Datei FPUZ80 in die FPU
– Anzeige des Installationsergebnisses
FMODE.EXE
– Initialisierung der Schnittstellen FCOM1 – FCOM8 durch Kommandoeingabe
I14BEN.EXE
– Prüfung, ob ein Anwenderprogramm INT14 benutzt.
Das Installationsergebnis nach Aufruf von FPULOAD wird entweder durch die Aufschrift

FPU running

für erfolgreiche Beendigung, oder durch

FPU-Hardware not found

angezeigt, was auf eine fehlerhafte Installation der FPU-Karte hindeutet.
Das Kommando FMODE hat dieselbe Syntax wie das DCP-Kommando MODE.
Entsprechend den im Abschnitt Anwendungsprogramme gemachten Aussagen zur Portabilität von Programmen mit INT14 sind Kommunikationsprogramme, die diesen Interrupt benutzen, grundsätzlich auch über die FPU lauffähig. Dazu muß jedoch die COM-Schnittstelle auf die gewünschte FPU-Schnittstelle (FCOMy) umgelenkt werden. Dies geschieht durch das Kommando

FMODE COMx:=FCOMy/REP

Das Standardfirmwareprogramm in der Datei FPUZ80 unterstützt die parallele Nutzung aller FPU-Schnittstellen (FCOM1 – FCOM4) über Modems (duplex/halbduplex) oder Nullmodem-Kabel.
Die Beispiele in den Bildern 1 und 2 zeigen die Nutzung von INT14 und INT60 in C- bzw. Assemblerprogrammen. Für die Prüfung, ob ein nicht selbst entwickeltes Programm den INT14 benutzt, ist vor dem Lauf des Nutzerprogramms das Programm I14BEN aufzurufen. Die Bildschirmausschrift

INT14

während des anschließenden Laufes des Nutzerprogramms zeigt die INT14-Nutzung durch das Programm an. Ein solches Programm ist auf der FPU lauffähig (vgl. Abschnitt Anwendungsprogramme).
Im VEB Leitzentrum für Anwendungsforschung (LfA) ist eine auf INT14 basierende CCSMV-Version entwickelt worden, welche die Kommunikation mit einer Vielzahl anderer Rechner und Betriebssysteme ermöglicht /1/. Für das Programmpaket uniNet /2/, /3/ ist im LfA eine Version erarbeitet worden, die dessen Nutzung auf einem PC mit FPU-Karte über INT60 ermöglicht. Außerdem ist in Ergänzung dazu im LfA das Programmpaket *TRACE* entwickelt worden, das es ermöglicht, alle Daten aufzuzeichnen, die die V.24-Schnittstelle eines PCs passieren. Damit wird die Programmierbarkeit der FPU für die Realisierung einer TRACE-Funktion genutzt.

```
#include "dos.h"
#define INT_FPU 0x14
   .
   .
main (...)
{
   .
   .
union REGS inregs;
union REGS outregs;
   .
   .
inregs.h.ah = 2;                          /* lesen Zeichen */
inregs.x.dx = 3;                          /* fcom2 */
int86(INT_FPU,&inregs,&outregs);          /* INT14-Aufruf */
if (outregs.h.ah == 0)                    /* lesen o.k. */
   printf("Empfangenes Zeichen: %c \n",outregs.h.al);
                                 /* Anzeige des empfangenen Zeichens */
   .
   .
}
```

***Bild 1* Nutzung des Interrupts 14 in C-Programmen**

```
   .
   .
MOV      AH,5               ; readmode 2 einstellen
MOV      DX,1               ; fcom2
INT      60h                ; Aufruf INT60
   .
   .
```

***Bild 2* Nutzung des Interrupts 60 in Assemblerprogrammen**

***Bild 3* Verbindung der zu analysierenden Schnittstelle COM1 mit der FPU**

Das Paket besteht aus den Komponenten:
TRACLOAD.EXE
– analog FPULOAD.EXE
– Laden (und Starten) der FPU–TRACE-Firmware aus der Datei TRACZ80 in die FPU
TRACRUN.EXE
– Starten der TRACE-Funktion
TRACDUMP.EXE
– Ausgabe des TRACE-Bereiches vom Hauptspeicher des PCs auf eine Datei.
Der erfolgreiche Ablauf des zuerst zu startenden Programms TRACLOAD wird durch die Bildschirmausgabe

TRACE INSTALLED

dokumentiert.
Die zu analysierende Schnittstelle ist mit einem Direktkabel mit der FCOM2-Schnittstelle zu verbinden. Die Verbindung, die vorher zwischen der zu analysierenden Schnittstelle und dem entfernten Gerät bestand, ist an die FCOM1-Schnittstelle anzuschließen (Bild 3).
Die Initialisierung der Schnittstellen FCOM1 und FCOM2 erfolgt, wie üblich, mit dem Kommando FMODE. Die Initialisierungswerte müssen denen der zu analysierenden Schnittstellen entsprechen.
Als nächstes ist das Programm TRACRUN zu starten, das mit der Aufschrift

TRACE RUNNING

endet.
Von diesem Zeitpunkt an werden alle Zeichen, die für die zu analysierende Schnittstelle zum Lesen bereitgestellt werden, mit den Vorzeichen << und alle über diese Schnittstelle gesendeten Zeichen mit >> protokolliert. Das Protokollierungsende wird durch das Zeichen ! markiert.
Die gesendeten und empfangenen Zeichen werden im Hauptspeicher des PCs von Adresse x000:0000 bis x000:2000 zyklisch abgelegt. (x entspricht der im Abschnitt Aufbau und Funktion erläuterten Einstellung der Sektoradresse für den Übergabe-/Übernahmebereich auf der Karte.) Dadurch kann eine zeitliche Abfolge der Ein- und Ausgabe über die zu analysierende Schnittstelle innerhalb eines von der anfallenden Datenmenge bestimmten Zeitintervalles verfolgt werden. Mit dem Programm TRACDUMP kann dieser Hauptspeicherbereich anschließend in einer Datei abgelegt werden, um ihn einer weiteren Analyse zuführen zu können. Soll FCOM1 oder FCOM2 analysiert werden, so ist die Schnittstelle mittels

FMODE FCOM1:=FCOM3/REP

auf FCOM3 umzuleiten und analog weiter zu verfahren.

Literatur

/1/ CCSMV – Einführungsschrift, VEB Leitzentrum für Anwendungsforschung Berlin 1988
/2/ Integration von Personalcomputern in Rechnernetze – uniNet/DCP. edv-aspekte, Berlin, 7 (1988) 3, S. 24
/3/ Heckert, R.; Peitz, J.: Inhouse-Kommunikation im Rechenzentrum mit uniNet. rechentechnik/datenverarbeitung, Berlin 25 (1988) 9, S. 10

KONTAKT

VEB Leitzentrum für Anwendungsforschung Berlin, Abt. R32, Jacques-Duclos-Straße 47/52, Berlin, 1156; Tel. 37803, App. 667

Call for Papers

Third International Workshop on COMPILER COMPILERS

October 22–26, 1990
Schwerin, GDR
Topics
- General requirements, properties, and theoretical aspects of compiler compilers
- Compilation for new computer architectures
- Development of (knowledge-based) generation tools
- Tools and metatools for software engineering

Members of the Programme Committee are from Canada, Finland, France, FRG, GDR, Hungary, Sweden, USSR.
Please contact
Academy of Sciences of the GDR, Institute of Informatics and Computing Technique, CC'90 – Organizing Committee, Mr. Albinus, Rudower Chaussee 5–6, Berlin, DDR-1199 *Albinus*

Multiprozessorsysteme mit U 8000

Dr. Frank Weicker, Jan Hamann
Wilhelm-Pieck-Universität Rostock
Sektion Technische Elektronik

Einführung

Die Leistungsfähigkeit prozeßnaher Signalverarbeitung läßt sich entscheidend dann erhöhen, wenn sich neben der Nutzung prozeßspezifischer Verbesserungen wie hohe Operationsgeschwindigkeit, verbesserter Befehlssatz und Einsatz hochspezialisierter Koprozessoren eine Parallelisierung der Aufgaben erreichen läßt. Von allen parallelverarbeitenden Systemen besitzen die Multiprozessorsysteme die größte Universalität /1/. Derartige Konfigurationen ermöglichen eine asynchrone, parallele Verarbeitung durch autonom arbeitsfähige Subsysteme. Diese Subsysteme sind ausgestattet mit allen Merkmalen eines Mikrorechners wie CPU, ROM, RAM, I/O-Interface u. ä. Ein denkbarer Anwendungsfall beispielsweise in der Stromrichterantriebstechnik ist die Aufteilung eines komplexen Informationsverarbeitungssystems in Meßwertrechner, Regelrechner, Ansteuerrechner usw. Diese können ihre Daten über eine gemeinsame Ressource, den Main-frame-RAM, austauschen (siehe Bild 1).

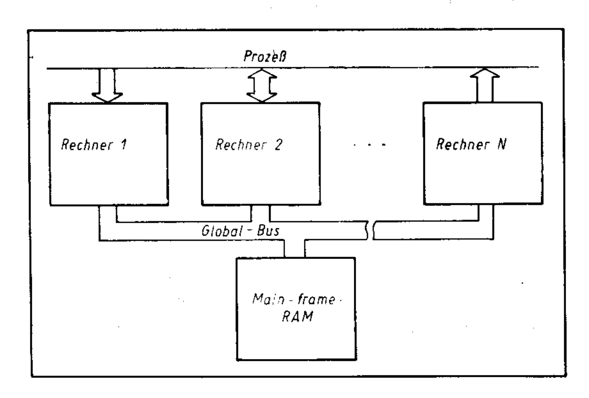


Bild 1 Prinzipbild eines Multiprozessorsystems

Neben der nicht zu unterschätzenden Aufgabe, den Informationsverarbeitungsprozeß sinnvoll zu parallelisieren, ist es erforderlich, den Zugriff der autonomen Subsysteme auf den Main-frame-RAM zu koodinieren.
Im folgenden wird gezeigt, welche Unterstützung der U 8000 anbietet, um einen koordinierten Main-frame-Ram-Zugriff erreichen zu können. Die vorgestellten Erkenntnisse sind Resultat eigener Applikationsarbeiten und stellen Basiswissen für den Aufbau von U 8000-Multiprozessorsystemen dar.

Prozessorseitige Voraussetzungen

Der Prozessor U 8000 besitzt effektive Hard- und Softwarehilfsmittel, die einen geregelten Zugriff der Einzelprozessoren auf eine gemeinsame Ressource unterstützen.

Hardware

Als Hardware bietet der U 8000 zwei diskrete Anschlüsse, den Eingang MULTI-MICRO-IN **/mI** sowie den Ausgang MULTI-MICRO-OUT **/mO**. **/mI** dient zum Erkennen der Anforderung eines Subsystems auf einen Datentransfer mit der gemeinsamen Ressource. Über **/mO** kann der Prozessor die Nutzung der gemeinsamen Ressource fordern. Tafel 1 zeigt die möglichen MULTI-MICRO-Zustände.

Tafel 1 MULTI-MICRO-Zustände des U 8000

Zustand	/mI	/mO	Interpretation
1)	High	High	Bezüglich des Ressourcenzugriffs sind alle Systeme inaktiv
2)	High	Low	Ressourcennutzung ist angemeldet, jedoch noch nicht bestätigt
3)	Low	Low	Ressourcennutzung ist angemeldet und bestätigt
4)	Low	High	Fremdes System nutzt Ressource

Software

Es gibt spezielle Befehle, die den Zustand von /mI testen sowie /mO gezielt manipulieren können. Diese vier Steuerbefehle sind privilegiert und somit nur im Systemmodus ausführbar:

MULTI-MICRO-BIT-TEST
Der Befehl **MBIT** hat keine Operanden. Er setzt den Sign-Status im Flagcodewort (FCW) je nach Signalpegel am Eingang /mI. Das Sign-Flag wird auf Null rückgesetzt, falls am Pin /mI ein High anliegt. Es wird auf Eins gesetzt, falls an /mI der Signalpegel Low ist. Andere Flags werden nicht beeinflußt. In einem Programm kann man mit diesem Befehl die Verfügbarkeit der Ressource prüfen, indem man nach Ausführung des Befehls das Sign-Flag testet. Nach /2/ könnte eine Befehlsfolge, die MBIT nutzt, folgendermaßen aussehen:

```
MBIT      ! teste /mI !
JP   MI,NA
```

(Befehlsfolge wird abgearbeitet, falls gemeinsame Ressource verfügbar ist.)

NA: (Befehlsfolge wird abgearbeitet, falls gemeinsame Ressource nicht verfügbar ist.)

MULTI-MICRO-REQUEST
Der Befehl **MREQ** hat als Operanden ein 16-Bit-CPU-Register, das eine Zeitkonstante enthält. Seine Ausführung beeinflußt das Zero- und das Sign-Flag. Folgende Vorgänge laufen ab:
Zuerst wird durch Abfrage des /mI-Einganges getestet, ob die gemeinsame Ressource benutzt wird. Ist /mI gleich Low, deutet das auf die Benutzung der Ressource durch ein Fremdsystem hin. Dann werden das Sign- und das Zero-Flag auf Null gesetzt. Der Ausgang /mO bleibt inaktiv, das heißt, es erfolgt keine Anforderung. Im Falle /mI ist gleich High, bedeutet das, daß die Ressource nicht belegt ist. Dann erfolgt die Anmeldung auf einen Transfer durch Aktivierung des Ausgangs /mO (/mO = Low).
Danach wird die im Register enthaltene Zeitkonstante alle sieben Takte um eins verringert. Damit erhält die externe Logik Zeit für den Durchlauf der Anforderung durch die Kette und die eventuelle Freigabe für den Zugriff auf die Ressource. Nachdem die Zeitkonstante zu Null dekrementiert wurde, wird der Eingang /mI getestet. Ist die Ressource besetzt (/mI = High), so wurde die Anforderung abgelehnt. (Dieser Fall kann eintreten, wenn ein höherpriorisiertes System gleichzeitig einen Transfer angemeldet hatte.) Das Zero-Flag wird dann auf Eins gesetzt, das Sign-Flag auf Null. Weiterhin wird die Anforderung zurückgenommen, indem /mO wieder auf High gesetzt wird. Somit ist die Befehlsausführung in diesem Fall beendet. Ist die Ressource dagegen verfügbar (/mI = Low), werden Sign- und Zero-Flag auf Eins gesetzt, und der Ausgang /mO bleibt Low. Der Zugriff kann beginnen. Die Befehlsausführung ist beendet.

MULTI-MICRO-RESET
Der Befehl **MRES** besitzt keine Operanden. Seine Ausführung beeinflußt kein Status-Flag. Die Ausführung des Befehls bewirkt, daß ein High am Ausgang /mO ausgegeben wird. Ein Prozessor gibt High an /mO aus, wenn er nicht auf die gemeinsame Ressource zugreifen will.

MULTI-MICRO-SET
Der Befehl **MSET** besitzt keine Operanden und beeinflußt kein Status-Flag bei seiner Ausführung. Wird MSET ausgeführt, so wird an /mO ein Low ausgegeben, womit die Nutzung der gemeinsamen Ressource angefordert wird.

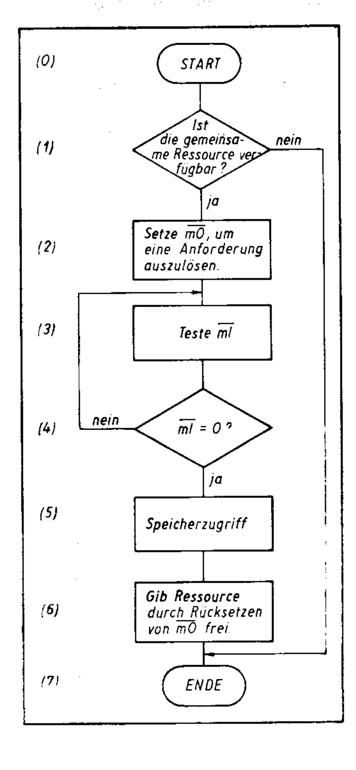


Bild 2 Flußdiagramm einer Anforderung

Handhabung der MULTI-MICRO-Steuerbefehle
Mit den soeben vorgestellten Befehlen ist es möglich, eine Anforderungsroutine aufzubauen. Das könnte zum Beispiel unter Nutzung der Befehle MBIT, MRES und MSET geschehen. Andererseits ist es auch denkbar, mit MREQ und MRES zu arbeiten.

Benutzt man die Befehle MBIT, MRES und MSET, so kann bei der Programmierung entsprechend dem in Bild 2 dargestellten Flußdiagramm vorgegangen werden. Mit MBIT wird die Verfügbarkeit der Ressource getestet. Falls die Ressource verfügbar ist, kann durch die Ausführung von MSET eine Anmeldung des Transfers über /mO veranlaßt werden. Danach wird mit MBIT der Eingang /ml getestet, um festzustellen, ob die Ressource frei ist. Ist sie verfügbar, kann der Prozessor auf den Main-frame-RAM zugreifen. Nach dem Transfer wird die Anforderung mit MRES aufgehoben.
Hat man sich für die Arbeit mit MREQ und MRES entschieden, so kann man eine kürzere Routine schreiben, und man hat somit offensichtlich die elegantere Form der Anforderung gewählt. Die in Bild 2 dargestellten Schritte (1) bis (3) führt der Befehl MREQ selbständig aus. Damit ist nach dem Speicherzugriff lediglich noch MRES für die Zurücknahme der Anforderung erforderlich.

Entwurf eines Multiprozessorsystems

Zum Aufbau eines Multiprozessorsystems sind die Signale /ml und /mO der Subprozessoren so zu verknüpfen, daß eine sinnvolle Kommunikation der Einzelsysteme erfolgen kann. Hierzu zählen das Durchschalten von Adreß-, Daten- und Steuerbus des anfordernden Subsystems auf einen den Mainframe-RAM steuernden Globalbus bei Ausschaltung von Zugriffskonflikten. Bezüglich der Hardwarekonfiguration der Subsysteme erscheint es sinnvoll, einen völlig identischen Aufbau und gleiche Bedingungen hinsichtlich ihrer Multiprozessor-Programmierbarkeit anzustreben.

MULTI-MICRO-DAISY-CHAIN

Kern des zu schaffenden Globalbusses sind 4 Signallinien, die als MULTI-MICRO-DAISY-CHAIN-Bus bezeichnet werden /3/ und die über jeweils eine Multiprozessorlogik die /ml- und die /mO-Linien aller Sub-CPUs manipulieren (siehe Bild 3).
Voraussetzung für das Verständnis der Multiprozessorlogik ist die Kenntnis der Signallinien /mAI, /mST, /mRQ, /mAO.

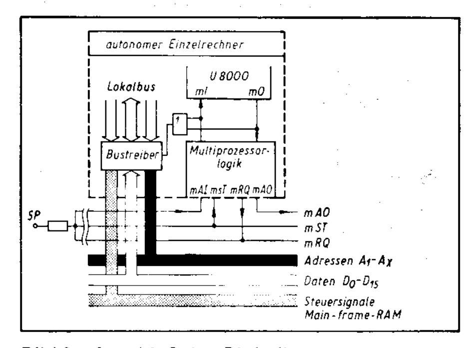


Bild 3 Anschluß des Globalbusses

/mAI: Der Eingang dient der Annahme eines Signals, das der anfordernden CPU anzeigt, ob ihre Anforderung durch die anderen CPUs der Daisy-Chain (Interruptprioritätenkette) akzeptiert worden ist. Für den Master ist diese Anfrage überflüssig.
/mST: dient der Feststellung des augenblicklichen Zustandes, in dem sich die Kette befindet.
/mRQ: Der Ausgang ist der Aussendung einer Anforderung auf einen Ressourcenzugriff vorbehalten.
/mAO: Der Ausgang wird zur Weitergabe einer an /mAI ankommenden Anforderung benutzt. Er bildet zusammen mit /mAI die Kettenschleife.

Bezugnehmend auf den Abschnitt Hardware kann den Multi-Micro-Zuständen folgendes Busverhalten zugeordnet werden (siehe auch Tafel 2):

Tafel 2 Angabe des logischen Verhaltens der Multi-Micro-Daisy-Chain-Buskonfiguration

/ml	/mO	mAI	/mAO	/mST	/mRQ
H	H	H	H	H	H
H	H	H	H	L	L
L	H	L	H	L	L
L	H	L	L	L	L
Prozessorsignale		Globalbus-Signale			

Zustand 1 (/ml = High, /mO = High)
Dieser Zustand ist dadurch gekennzeichnet, daß alle Systeme inaktiv sind. Da kein Prozessor versucht, einen Transfer über den Signalausgang /mO und damit über /mRQ anzumelden, ist auch der Status der Kette /mST High. Ein auf High liegendes /mRQ führt dazu, daß die Signallinien /mAI und /mAO gleich High sind. Ein High-Signalpegel an /mST zieht entsprechend der Funktion des Schaltkreises 74L157 ein /ml = High nach sich.

Zustand 2 (/ml = High, /mO = Low)
In diesem Zustand befindet sich ein System, wenn es gerade eine Anforderung auf einen Transfer aussendet. Über den CPU-Ausgang /mO wird ein Low ausgegeben, das ein Low-aktives Signal /mRQ erzwingt. Dadurch wird auch das den Status der Kette kennzeichnende Signal /mST = Low. Gleichzeitig wird an /mAO ein High erzeugt, womit die Anforderung durch niederpriorisierte CPUs blokkiert wird. Zunächst ist das Signal /mAI noch High, was ein High am CPU-Eingang /ml zur Folge hat.

Zustand 3 (/ml = Low, /mO = Low)
Wird nach einer gewissen, durch die Länge der Kette bedingten Laufzeit die Signalleitung /mAI = Low, so wird ein Low an /ml erzeugt. Damit erkennt die CPU, daß ihre Anforderung durch alle in der Kette befindlichen CPUs akzeptiert wurde.

Zustand 4 (/ml = Low, /mO = High)
Dieser Zustand ist durch die Benutzung der Ressource durch ein Fremdsystem charakterisiert. Eine fremde Anforderung über /mRQ bringt aufgrund der Wired-AND-Verknüpfung ein Low an allen Ausgängen /mRQ des Daisy-Chain-Busses mit sich. Das Signal /mST wird ebenfalls Low, was ein Low an /ml bewirkt. Je nach Priorität des Einplatinenrechners gegenüber dem aktiven System wird an dessen Eingang /mAI ein Low anliegen (Teilsystem ist höherpriorisiert) oder ein High (Teilsystem besitzt eine niedere Priorität).
Ein Low an /mAI ruft seinerseits ein Low am Ausgang /mAO des Interfaces hervor. Ein High an /mAI hat ein High an /mAO zur Folge.

Multiprozessorlogik

Bild 4 zeigt die Schaltung zur Realisierung einer Multiprozessorlogik, die leicht auf jedem Einzelrechner untergebracht werden kann.

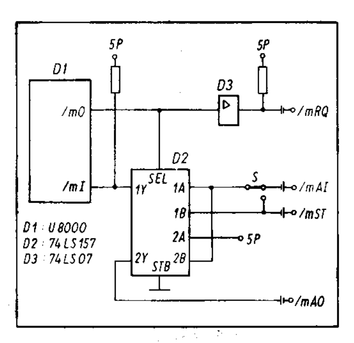


Bild 4 Multiprozessorlogik

Es wird empfohlen, den Vierfach-2-Bit-Datenselektor 74LS157 oder 74LS257 (open collector) zu nutzen und nicht wie in /3/ zu versuchen, die Schaltung mit Hilfe von Gattern „nachzuempfinden" (akute Fehlerquelle!). Soll ein Einzelrechner die Masterfunktion übernehmen, ist mittels Schalter das mAI-Signal auf mST zu legen.

An dieser Stelle soll darauf hingewiesen sein, daß auch andere Prozessoren in das Konzept einbezogen werden können, wenn das ml-mO-Signalverhalten durch zwei Portlinien und entsprechende Bediensoftware realisiert wird.

Die geforderte Konfliktfreiheit des Zugriffs auf den Main-frame-RAM läßt sich dadurch erzielen, daß über Bustreiber nur jeweils *der* Bus des Einzelrechners auf den globalen Bus durchgeschaltet wird, dessen Zugriffsanforderung bestätigt wurde. Das erforderliche Busfreigabesignal /OE der Treiber ist durch ODER-Verknüpfung der /ml- und /mO-Linien generierbar (Bild 3).
Welche RAM-Steuersignale zur Verfügung gestellt werden, ist von der Systemphilosophie abhängig. Eine vorteilhafte Lösung besteht bei Verwendung der nichtsegmentierten Version U 8002 darin, das /CS-Signal des Main-frame-RAMs aus der Adreßdekodierung für den Speicherblock der oberen 32-K-Wörter zu gewinnen. Wird darüber hinaus ein wahlweiser Wort-/Byte-Zugriff auf die Ressource gewünscht, sind folgende Signale auf den Globalbus zu führen: /CS-RAM, AO, B/W, R/W.

Beispiel für ein MULTI-MICRO-Anforderungsprotokoll

Da man die Arbeit mit MREQ und MRES bevorzugen wird, befaßt sich dieser Abschnitt etwas ausführlicher mit den bei Anmeldung durch MREQ möglicherweise eintretenden Fällen. Zur Verdeutlichung der ablaufenden Vorgänge dient Bild 5. Wird versucht, mit

Fortsetzung auf Seite 339

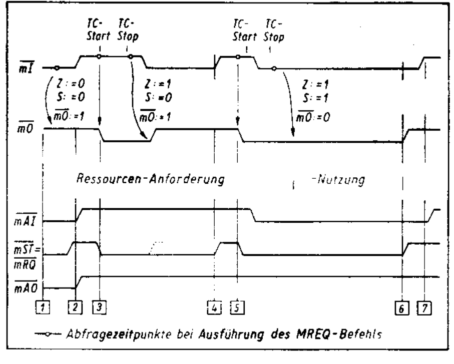


Bild 5 Anforderungsprotokoll

Einführung in Forth-83

Teil 5

Dr. Hartmut Pfüller (Leiter),
Dr. Wolfgang Drewelow, Dr. Bernhard Lampe
Ralf Neuthe, Egmont Woitzel
Wilhelm-Pieck-Universität Rostock,
Sektion Technische Elektronik

5. Die qualitative Erweiterung von Forth

Beim Vergleich mit anderen Programmiersprachen fällt auf, daß im Kern von Forth keine Definitionsworte für Datenstrukturen wie Felder oder Records enthalten sind. Die Bedeutung von Datenstrukturen ist aber für die Programmierung unbestritten, so daß die Frage steht, wie solche Probleme in Forth rationell zu behandeln sind.
Anstatt eine mehr oder weniger fest vorgefertigte Menge von starren Datentypen vorzugeben, stellt Forth dem Programmierer Werkzeuge zum Eigenbau von Datenstrukturen zur Verfügung. Er kann damit seine Datenstrukturen für das Problem maßschneidern und muß nicht das Problem solange umformen, bis es in eine angebotene Datenstruktur paßt.
Zu dieser für Forth typischen Vorgehensweise gehört auch die Möglichkeit, spezielle Werkzeuge zu schaffen, die eine rationelle und problemnahe Behandlung der selbsterzeugten Datenstrukturen sichern. Welchen Weg schlägt Forth ein, um dieses Ziel zu erreichen? Das Prinzip ist vergleichbar mit der Vorgehensweise bei der im Teil 3 beschriebenen Ausgabeformatierung. Auch die komplexen Definitionsworte wie **:** oder **VARIABLE** basieren auf elementaren Bestandteilen, die dem Programmierer sämtlich einzeln zugänglich sind und die zur Konstruktion neuer definierender Worte benutzt werden können.

5.1. Die Verwaltung des Wörterbuchspeichers

5.1.1. Funktionen

Eine Orientierung darüber, wo für den nächsten Wörterbucheintrag freier Speicherplatz ist, erhält man dadurch, daß das Wort **HERE** die entsprechende Adresse zum Stapel schafft. Wenn diese Adresse in einer (nicht standardisierten) Variablen **DP** (Directionary Pointer, deutsch: Wörterbuchzeiger) geführt wird, könnte HERE so implementiert sein:

```
: HERE ( ===> addr) DP @;
```

Um das Wörterbuch am Ende zu verlängern, kann das Wort **ALLOT** benutzt werden. ALLOT erwartet eine Zahl auf dem Datenstapel und reserviert dem Wert entsprechend viele Bytes für das Wörterbuch von der durch HERE bezeichneten Stelle ab. Der Inhalt der mit ALLOT reservierten Speicherplätze wird nicht gesetzt, er bleibt so, wie er sich zufällig bis zu diesem Zeitpunkt ergeben hatte. Systemintern könnte ALLOT etwa so definiert sein:

```
: ALLOT (n ===>) DP +! ;
```

Ebenfalls eine Vergrößerung des Wörterbuchs erreicht man mit den beiden Worten

, C,

Das Wort **,** schreibt die auf dem Datenstapel befindliche 16-Bit-Zahl ans Wörterbuch hintan; das Wörterbuch wird damit um 2 Byte länger, wobei der Inhalt dieser 2 Byte durch die vom Stapel geholte Zahl gesetzt wird. Entsprechend vergrößert **C,** das Wörterbuch um genau ein Byte, wobei die 8 Bit, die dessen Inhalt bestimmen, auch wieder als Wert vom Stapel geholt werden. Die beiden Worte könnten im System beispielsweise so definiert sein:

```
: , (16b ===>) HERE 2 ALLOT ! ;
: C, (8b ===>) HERE 1 ALLOT C! ;
```

Beim Zugang zur physischen Ebene müssen von Fall zu Fall spezifische Schaltkreiseigenschaften berücksichtigt werden. Angenommen, der Programmierer wünscht, ins Wörterbuch eine Tabelle von Zeigern zu Interruptserviceroutinen einzutragen, und der Prozessor verlangt, daß die Tabelle bei einer durch 8 teilbaren Adresse beginnt. Die Variable DP kann dann durch eine Wortfolge wie

```
HERE NEGATE 8 MOD ALLOT
```

auf die nächste durch 8 teilbare freie Adresse des Wörterbuchs gesetzt werden. Mittels solcher und ähnlicher Techniken kann man beispielsweise die Interruptbehandlung vollständig von Forth aus organisieren.

5.1.2. Vektoren und Tabellen

Die genannten Mittel gestatten bereits die Vereinbarung von Datenbereichen im Wörterbuch. Beispielsweise kann durch die Sequenz

```
VARIABLE ZAEHLERFELD 8 ALLOT
```

ein Feld für 5 Werte von je 16 Bit im Wörterbuch reserviert werden; ein Wert direkt durch **VARIABLE** und die restlichen vier Werte durch **8 ALLOT. VARIABLE** legen bei der Definition von **ZAEHLERFELD** dessen Laufzeitverhalten fest: Beim Aufruf von ZAEHLERFELD wird die Basisadresse (die Adresse der 0. Komponente) auf den Datenstapel gelegt. Die übrigen Komponenten sind durch den Offset von 2, 4, 6, 8 adressierbar, für Index 3 etwa so:
ZAEHLERFELD 6 +
Die Initialisierung aller fünf Zähler mit Null kann nach

```
: RUECKSETZEN ( 'feld ===> )
  5 0 DO 0 OVER ! 2+
    LOOP DROP ;
```

durch die Wortfolge
ZAEHLERFELD RUECKSETZEN
gemeinsam ausgeführt werden. Durch die Zusammenfassung von Zählern in einem Feld können solche Operationen vereinfacht werden, die alle Zähler des Feldes betreffen. Dazu gehören neben der Initialisierung beispielsweise Vergleiche der Zählerstände oder deren Auswertung und Darstellung.
Im obigen Beispiel ist es eventuell nachteilig, daß die Initialisierung nur für Felder der festen Dimension 5 arbeitet. Will man Felder variabler Länge in ähnlicher Weise behandeln, so muß die zugeordnete Feldlänge ermittelt werden können. Das läßt sich erreichen, wenn die Felddimension mit abgespeichert wird (beispielsweise vor dem Feld selbst). Entsprechend der oben stehenden Variante könnte man

```
VARIABLE ZAEHLERFELD 10 ALLOT
5 ZAEHLERFELD !
```

schreiben. Noch einfacher ist jedoch die Konstruktion, wenn man als Definitionswort für den Wörterbucheintrag das (primitivere) Definitionswort **CREATE** verwendet. Es wirkt wie VARIABLE, reserviert selbst aber nach dem Namen noch keinen zusätzlichen Speicherplatz für Daten. Die Folge

```
CREATE ZAEHLERFELD 5 , 10 ALLOT
```

spiegelt die Struktur des Feldes gut wider; CREATE legt das Laufzeitverhalten von ZAEHLERFELD fest. Das mit

```
: INITIALISIEREN ( 'feld ===> )
  DUP @ 0 DO 2+ 0 OVER !
    LOOP DROP ;
```

definierte Wort initialisiert so ein Feld variabler Länge.
Fließpunktarithmetik und transzendente Funktionen sind nicht Bestandteil des Forthkerns, weil sich die Anforderungen der Anwender an Genauigkeit und Verarbeitungsgeschwindigkeit stark unterscheiden. Außerdem lassen sich viele Probleme mit ganzen Zahlen lösen, wenn man auf die (gefährliche) Bequemlichkeit des Rechnens ohne vorherige Abschätzung verzichtet. Als Beispiel soll die Funktionsberechnung unter Verwendung von Funktionstafeln erläutert werden, wobei man bei geeigneter Skalierung im Bereich der ganzen Zahlen bleibt.
Eine mit

```
CREATE SINUSTAFEL
     0 ,  175 ,  349 ,  523 ,  698 ,
   872 , 1045 , 1219 , 1392 , 1564 ,
  1737 , 1908 , 2079 , . . .
   .
   .
   .     9976 , 9986 , 9994 , 9999 ,
 10000 ,
```

erzeugte Tafel enthält die mit 10000 skalierten Sinuswerte zu den Winkeln von 0, 1, . . ., 90 Grad. Hier wird die Zweckmäßigkeit der Vergabe des Namens **,** für die direkte Eintragung ins Wörterbuch erkennbar. Ein Leser des Programmteils SINUSTAFEL muß nicht unbedingt mit dem an das Komma gebundenen Kompilationsvorgang vertraut sein; es genügt, die im gewöhnlichen Sprachgebrauch übliche Bedeutung des Kommas als Trennsymbol zu kennen. Durch die ähnlich geschickte Wahl der Namen kann der Programmierer die Lesbarkeit seiner Programme günstig beeinflussen. Das Aufsuchen des passenden Sinuswertes geschieht nach

```
: SINUS ( n ===> sin[n] )
  2 * SINUSTAFEL + @ ;
```

in der Form
45 SINUS . (cr) 7071 ok

Unter Ausnutzung der Periodizität können beliebige Winkel auf den angegebenen Argumentbereich zurückgerechnet werden. Durch einen Suchprozeß innerhalb der Tabelle ist auch die Arkussinusfunktion implementierbar. Das oben definierte Wort **SINUS** stellt die primitivste Form des Zugriffs auf die Tabelle dar. Durch lineare Interpolation ließen sich auch Zwischenwerte berechnen.

5.2. Definition von Wortklassen

Wenn mehrere Objekte die gleiche oder eine ähnliche Struktur haben, kann man, wie am Beispiel der Initialisierung deutlich wurde, Operationen so erzeugen, daß sie über allen Objekten einer solchen Familie ausführbar sind. Wünschenswert wäre nun, auch die Kompilationsvorschrift für zur gleichen Familie gehörende Objekte nur einmal erklären zu müssen. Das spart Programmspeicherplatz und erhöht die Übersichtlichkeit. Mit der einmal erzeugten Form können dann viele gleichartige Objekte „gegossen" werden.
Die Konstruktion einer „Gußform" wird ermöglicht, weil auch Definitionsworte (z. B. CREATE) selbst wieder in Definitionen verwendet werden dürfen. Auf diese Art kann der Anwender dann eigene Definitionsworte definieren, die bei ihrer Ausführung selbst wieder neue Worte eines (vom Anwender) bestimmten Typs erzeugen. So können mit einem vom Programmierer neu geschaffenen Definitionswort spezielle Eigenschaften für eine ganze Klasse von neuen Worten fixiert werden. Die zweckmäßige Festlegung von problemgerechten Wortklassen kann in hohem Maße die Qualität eines Programms bestimmen. Sie verlangt eine gründliche Analyse der Problemstellung, insbesondere unter dem Gesichtspunkt der Zerlegung in ähnlich geartete Teilprobleme. Nach Erklärung des Datentyps

```
: FELD ( n ===> )
  CREATE DUP , 2 * ALLOT ;
```

beispielsweise würde durch

5 FELD ZAEHLERFELD (===> addr)

ein Zählerfeld der Dimension 5 ins Wörterbuch eingetragen werden. FELD ist nun das Definitionswort für einen vom Programmierer neu geschaffenen Datentyp. Das Laufzeitverhalten von ZAEHLERFELD wird wiederum durch CREATE festgelegt. Nach Aufruf von ZAEHLERFELD würde die Parameterfeldadresse des ZAEHLERFELDes auf den Datenstapel gelegt werden; dort war bei der Definition durch FELD die Dimension eingetragen worden.

5.2.1. Fadencodeniveau

Mitunter wird es sinnvoll sein, wenn nach dem Aufruf eines mit FELD erzeugten Objektes nicht die Adresse der Dimension, sondern die Adresse des ersten Feldelements zum Stapel geliefert wird. Das Laufzeitverhalten, das dem neudefinierten Wort durch CREATE mitgegeben wurde, müßte zu diesem Zweck anders programmiert werden können. Speziell für diesen Zweck ist das Wort **DOES>** vorgesehen. Die in der Konstruktion

: name . . . CREATE . . . DOES> . . . ;

nach DOES> stehenden Worte werden bei Aufruf jedes mit *name* definierten Objektes ausgeführt. Dabei ist es wichtig zu wissen, daß unmittelbar vorher die Parameterfeldadresse des Objekts auf dem Stapel bereitgelegt wird. Die auf DOES> folgenden Worte können dann diese Information über den Beginn des Parameterfeldes verwenden, um damit ein gewünschtes Laufzeitverhalten des Objekts zu erzeugen.
Eine neue Klasse von Datenfeldern, deren Abkömmlinge zwar ihre Dimensionen enthalten, aber beim Aufruf die Adresse ihrer nullten Komponente auf den Stapel legen, wird mit

```
: VEKTOR ( n ===> )
  CREATE DUP , 2* ALLOT
  DOES> ( ===> 'x0 ) 2+ ;
```

beschrieben. Beim Stapelkommentar nach DOES> muß man immer berücksichtigen, daß unmittelbar vor dem Abarbeiten der DOES>-Passage noch die Parameterfeldadresse des definierten Objekts obenauf gelegt wird. Sie sollten etwas Zeit investieren, um sich klarzumachen, welche Konsequenzen es hat, daß die Passage hinter DOES> *nicht* bei der Ausführung von VEKTOR ausgeführt wird. Vielmehr sorgt DOES> dafür, daß bei jeder Definition eines Objekts mittels **VEKTOR** das Codefeld dieses Objekts mit einem Verweis zur DOES>-Passage überschrieben wird. Die Worte nach DOES> werden damit genau bei jedem Aufruf eines Objekts abgearbeitet. Dadurch kommt auch die Familieneigenschaft aller mit demselben Definitionswort erzeugten Objekte zustande. Der Zugriff auf Komponenten von mit VEKTOR definierten Vektoren wie

5 VEKTOR ZAEHLERFELD (===> addr)
ließe sich durch

```
: KOMPONENTE ( 'x n ===> 'xn)
  2* + ;
```

bewerkstelligen. So wird durch
ZAEHLERFELD 3 KOMPONENTE @
der Inhalt des Elements mit dem Index 3 auf den Datenstapel gelegt.
Die Konstruktion **CREATE . . . DOES>** macht es möglich, die Kompilationsphase für Objekte einer Wortklasse vollständig von der Phase der Ausführung getrennt zu betrachten und zu programmieren. Der Programmierer kann die aus seiner Sicht insgesamt erforderlichen Aktionen zerlegen und der jeweils entsprechenden Phase zuordnen (für den Kompilationsvorgang hinter CREATE und für den Ausführungsvorgang hinter DOES>).
Zur Erläuterung des Gebrauchs von Wortklassen soll nun ein Beispiel aus Teil 2 des Kurses etwas modifiziert behandelt werden: In einer Variablen MATERIAL stehe (wie auch damals) die Adresse des Zählers für Teile des gerade aktuellen Materials. Die Reservierung von Speicherplatz für einen materialabhängigen Zähler und die Bereitstellung seiner Adressen bei Nennung des Materials werden jetzt in dem Definitionswort

```
: MATERIAL:
  VARIABLE ( installiert Zähler)
  DOES> MATERIAL ! ( lädt Zähler) ;
```

zusammengefaßt. Nach den Definitionen
MATERIAL: HOLZ
MATERIAL: PLAST
und den „Verschönerungen"

```
: DAZU ( n addr ===> ) +! ;
: ? ( addr ===> ) @ . ;
```

sind die in Teil 2 benutzten Aktionen wie
7 HOLZ TEILE DAZU
PLAST TEILE ?
wieder verwendbar.
Will man verschiedenartige Teile nach Material sortiert zählen, so kann man für jedes Material mehrere „Körbe", das heißt ein Zähler*feld* bereitstellen. Der zum jeweiligen Teil gehörende Korb ist durch einen Offset zum Anfang des Zählerfeldes des entsprechenden Materials erreichbar (Bild 5.1).
Das Definitionswort TEIL: speichert während der Definition eines Teils den vom Programmierer zugeordneten Offset n verdoppelt ab

```
VARIABLE  MATERIAL

:  TEIL:          ( n ===> )
   2  *  CONSTANT
         DOES>    ( ===> korbadresse)
            @   MATERIAL  @  +  ;

2  CONSTANT   #TEILE
0  TEIL:      KEILE
1  TEIL:      STIFTE

:  MATERIAL:  ( ===> )
   CREATE  #TEILE   0  DO  0  ,  LOOP
   DOES>      ( ===> )   MATERIAL  !  ;

MATERIAL:  HOLZ
MATERIAL:  PLAST
```

Bild 5.1 Beispiel zu CREATE . . DOES>

und bildet bei Nennung des jeweiligen Teils die Adresse des zugehörigen materialrichtigen Zählers. Damit ist die Schreibweise

HOLZ 12 KEILE DAZU 60 STIFTE DAZU PLAST STIFTE ?

möglich. Wenn für #TEILE vorsorglich ein etwas größerer Wert vereinbart wird, können mittels TEIL: auch noch nachträglich weitere Arten von Teilen definiert werden. Die Konstruktion CREATE . . . DOES> ist einfach, leistungsfähig und ungewöhnlich. Wir empfehlen Ihnen, eigene Experimente mit betont unkomplizierten Schritten zu beginnen. Eventuell sind in der ersten Zeit erläuternde Hilfsausschriften nützlich, wie im folgenden Beispiel für ein Definitionswort, das Objekte erzeugen kann, die nichts anderes tun, als ihre eigenen Aktivierungen mitzuzählen:

```
: ZAEHLER ." vor Definition "
  CREATE ." nach Definition " 0 ,
  DOES> ." bei Aktivierung"
    1 OVER +! @ . ;
```

Nach der Erzeugung eines Objekts mit
ZAEHLER KLICK
zeigt KLICK bei jeder Aktivierung an, der wievielte Aufruf das war.

5.2.2. Mikrocodeniveau

Alternativ zum Fadencode kann der Ausführungsteil auch in Maschinencode notiert werden. Für diesen Fall ist das Wort **;CODE** anstelle von **DOES>** zu benutzen. Die auf ;CODE folgenden Worte müssen dann Be-

standteil des Forth-Assemblers sein, der vorher geladen worden sein muß. Falls kein Assembler verfügbar ist, kann auch der von Hand übersetzte Maschinencode direkt mit Hilfe von **,** oder **C,** eingetragen werden. Man wird ;CODE benutzen, wenn man auf die größere Rechengeschwindigkeit von Assemblerprogrammen angewiesen ist. Zum Beispiel könnte man die Zugriffszeit auf Werte in Tabellen verkürzen, wenn man sie mit dem Definitionswort nach Bild 5.2 erzeugt. Der Programmierer braucht auch hier einen Zugang zum Parameterfeld der letztlich zu definierenden Objekte. Während DOES> hierfür die Parameterfeldadresse (PFA) auf dem Stapel hinterläßt, regelt sich nach ;CODE der Zugang zur PFA des definierten Objekts über das WA-Register der virtuellen Forthmaschine. Welches Prozessorregister in einer konkreten Implementation dafür verwendet wird, muß der zugehörigen Dokumentation entnommen oder anderweitig in Erfahrung gebracht werden. Selbstverständlich hat man sich mit der hier vorgeführten Version für das Definitionswort TAFEL auf den Prozessor Z 80 und meist auch noch auf einen bestimmten Assembler festgelegt.

Nach der Vorbereitung

10000 9999 . . . 175 0
91 TAFEL SINUS

kann die Sinusfunktion für einen bestimmten Winkel wieder mit

n SINUS

ermittelt werden. Solche Tafeln sind ein sehr bequemes und rechenzeiteffektives Verfahren, um Funktionen implementieren zu können.

```
:  TAFEL        ( xn-1 . . . xi x0 n ===> )
   CREATE
        0  DO  ,  LOOP
   ;CODE        ( i ===> xi )
   DE       INC,                        ( WA-Register auf PFA justieren)
   HL       POP,                        ( Index i vom Stapel holen)
   HL HL    ADD,  DE HL  ADD,           ( Elementadresse berechnen)
   (HL)  E  LD,   HL     INC,           ( Element nach DE laden)
   (HL)  D  LD,
   DE      PUSH,                        ( Element xi zum Stapel)
   NEXT  #  JP,                         ( zurueck zum Adressinterpreter)
   END-CODE
```

Bild 5.2 Beispiel zu CREATE...;CODE

5.3. Compilererweiterungen

5.3.1. Vorrangworte

Im Teil 2 des Kurses (s.MP 5/89) wurden Strukturierungsworte vorgestellt. Sie dienen im Kompilationsmodus zum Aufbau eines möglichst laufzeitrationellen Fadencodes und werden deshalb innerhalb einer Definition *ausgeführt,* während „normale" Worte im Gegensatz dazu in Fadencode *kompiliert* werden. Soll ein bestimmtes vom Programmierer definiertes Wort diese besondere Eigenschaft erhalten, so daß es trotz des Kompilationsmodus ausgeführt wird, so ruft man nach dessen Definition das Wort **IMMEDIATE** (deutsch sinngemäß: unverzüglich) auf. Das zuletzt definierte Wort wird dadurch zu einem *Immediate*-Wort oder *Vorrangwort.*

Die Ausführung von Vorrangworten wird gewissermaßen in die Kompilationszeit vorgezogen.

Das bisher betont einfache Prinzip der Kompilation von neuen Worten durch Zusammenstellung von Worten im Quelltext und durch – dafür jeweils eins zu eins – deren Codefeldadressen im Kompilat kann mittels der Vorrangworte durchbrochen werden, weil sie in der Regel nicht ins Wörterbuch gelangen und stattdessen sogar andere Worte ins Wörterbuch kompilieren können. Das würde der Programmierer zunächst gar nicht bemerken, wenn er von anderen Programmiersprachen her gewöhnt ist, daß ihm der Zugang zur Gestaltung des Kompilats in der Regel verschlossen bleibt.

Der versierte Forthprogrammierer wird von Fall zu Fall wünschen, Vorrangworte nicht zur Kompilationszeit arbeiten zu lassen, sondern sie entgegen ihrem Normalverhalten trotzdem zu kompilieren. Dafür gibt es die Möglichkeit, die Vorrangeigenschaft fallweise explizit zu unterdrücken.

Mit dem Wort **[COMPILE]** kann man die Kompilation eines Wortes erzwingen; die eventuelle Vorrangeigenschaft wird unterdrückt. Die erzwungene Kompilation betrifft nur das unmittelbar auf [COMPILE] folgende Wort. Damit [COMPILE] diese Wirkung erzielen kann, ist es selbst als Vorrangwort erklärt.

Als Gegenstück zur vorgezogenen Ausführung von Worten ist auch eine aufgeschobene Kompilation möglich. Das erreicht man durch eine Wortfolge der Form

: yyy . . . COMPILE xxx . . . ;

Der Effekt ist der, daß yyy bei seiner Ausführung eine Kompilation von xxx durchführt (also die Eintragung der Codefeldadresse von xxx ans Wörterbuchende). Während der Definition von yyy werden die CFAs von **COMPILE** und anschließend von xxx ganz normal ins Wörterbuch eingetragen. Beim Aufruf von yyy sorgt dann COMPILE dafür, daß die CFA von xxx noch einmal an das aktuelle Ende des Wörterbuchs kompiliert wird. Typisch wird COMPILE in Vorrangworten verwendet, wie wir in einem Beispiel im nächsten Abschnitt zeigen werden.

5.3.2. Programmverzweigungen

Im Abschnitt 2.3 wurden Worte gezeigt, mit denen Programme nach den Regeln der strukturierten Programmierung gegliedert werden können. Bei der Erzeugung *eigener* Verzweigungsstrukturen ist es zuweilen effektiver, sie aus elementaren Bestandteilen neu aufzubauen, anstatt die aus dem Anwenderwortschatz geläufigen vorgefertigten Strukturierungsworte zu verwenden.

Zu den elementaren Bestandteilen der Verzweigungsstrukturen gehören die Worte

>MARK >RESOLVE <MARK <RESOLVE

Mit ihnen werden zur Kompilationszeit Sprungadressen in Verzweigungsstrukturen verwaltet. Es gehören jeweils eine Markierung (MARK) und eine Auflösung (RESOLVE) zusammen. Mit MARK wird eine Marke (Adresse) auf den Datenstapel gelegt, zu der später von derjenigen Position des Wörterbuchzeigers aus gesprungen werden muß, die bei Nennung von RESOLVE gerade aktuell war (oder wird). Die Richtung der Sprünge ist an den Namen MARK und RESOLVE durch < für rückwärts und > für vorwärts abzulesen.

Besonders einfach ist die Eintragung von Rückwärtssprüngen durch

: <MARK (===> addr) HERE ;
: <RESOLVE (addr ===>) , ;

nachvollziehbar. Bei Vorwärtsreferenzen wird die Zieladresse erst zu einem späteren Zeitpunkt ermittelt, so daß der Speicherplatz erst reserviert und später die Adresse dort eingetragen wird. Auch hier läßt sich der möglichen Implementierung

: >MARK (===> addr)
HERE 2 ALLOT ;
: >RESOLVE (addr ===>)
HERE SWAP ! ;

die Arbeitsweise dieser Worte entnehmen. Da die Worte nur die Eintragung der Sprungadressen leisten, müssen sie noch mit Sprungbefehlen und Sprungbedingungen verknüpft werden (vergl. BRANCH, ?BRANCH in Teil 4 des Kurses).

Die Sprungadreßberechnungen sind so angelegt, daß sie sich verschachteln lassen. Damit dem Programmierer dabei keine Flüchtigkeitsfehler unterlaufen, prüfen die meisten Forth-Systeme während der Kompilation die Vollständigkeit der Verzweigungsstrukturen. Oft wird dazu bei Eröffnung der Verzweigung über die Adresse eine Kennzahl gelegt, deren Anwesenheit im Schließbefehl vor Ausführung der Adreßverknüpfung überprüft wird. Auf solche einfachen Sicherungsmaßnahmen sollte man bei eigenen Kompilationsworten nicht verzichten. Man kann eigene Kompilationsworte gerade mit dem Ziel schaffen, während der Kompilation weitgehend die Richtigkeit des Programms zu überprüfen.

Als Beispiel für neue Kompilationsworte soll eine inzwischen weit verbreitete Fallkonstruktion betrachtet werden. Sie verallgemeinert die Struktur **IF . . . ELSE . . . THEN** auf eine beliebige Anzahl von Fällen. Welcher Zweig des Programms abgearbeitet wird, soll sich aus dem Wert einer zur Abarbeitungszeit auf dem Datenstapel liegenden Zahl (Fallnummer) ergeben. Die Struktur wird in der Weise

```
: ... CASE n1 OF ... ENDOF
           n2 OF ... ENDOF
            .  .  .  .
           nk OF ... ENDOF
  ... ENDCASE ... ;
```

benutzt. Die vorkommenden Worte können wie in Bild 5.3 definiert werden. Alle vier definierten Strukturierungsworte werden jeweils durch IMMEDIATE zu Vorrangworten erklärt; mit COMPILE werden die erforderlichen Wörterbucheinträge erzwungen.

Die Kompilation eines die **CASE**-Konstruk-

```
:  CASE   ( ===> addr 5 ; Strukturanfang )
   CSP  @  SP@  CSP  !  5  ;           IMMEDIATE

:  OF     ( 5 ===> addr 6 ; Zweiganfang )
   5  ?PAIRS  ( Struktursicherung )
   COMPILE  OVER    COMPILE  =
   COMPILE  ?BRANCH
   >MARK   COMPILE DROP   6  ;         IMMEDIATE

:  ENDOF  ( addr1 6 ===> addr2 5 ; Zweigende)
   6  ?PAIRS    ( Struktursicherung )
   COMPILE  BRANCH
   >MARK  SWAP  >RESOLVE  5 ;          IMMEDIATE

:  ENDCASE  ( addr addr1 ... addrk ===>)
   5  ?PAIRS    ( Struktursicherung )
   COMPILE  DROP
   ( Spruenge ans Strukturende nachtragen:)
     BEGIN  SP@  CSP  @  -  WHILE  >RESOLVE
     REPEAT
   CSP ! ; ( CSP wiederherstellen)  IMMEDIATE
```

Bild 5.3 Codierung einer CASE-Verzweigung

tion enthaltenden Wortes soll nun kurz verfolgt werden:
Das Wort CASE eröffnet die Struktur, indem es den Wert der Variablen **CSP** auf den Stapel rettet, den aktuellen Stand des Stackpointers für einen späteren Vergleich in der Variablen CSP speichert und die willkürlich gewählte Kennzahl 5 zur späteren Kontrolle der Vollständigkeit der Struktur auf den Stapel legt. Danach wird die Fallnummer ins Wörterbuch kompiliert. Das darauf folgende **OF** kontrolliert die richtige Reihenfolge innerhalb der CASE-Konstruktion, indem die durch CASE auf dem Stapel hinterlassene 5 abgefragt wird. Falls **?PAIRS** nicht bekannt ist, kann man vorläufig

: ?PAIRS (n1 n2 ===>)
– IF ." falsche Struktur " THEN ;

definieren. Normalerweise sollte man allerdings abbrechen, wenn Strukturfehler erkannt worden sind. Im Teil 6 des Kurses wird auf Möglichkeiten zur Fehlerbehandlung eingegangen.
Anschließend werden mit der Wortfolge **OVER = ?BRANCH**, der Platz für die Sprungadresse und **DROP** kompiliert. Zum Abschluß von OF wird die Kennzahl 6 auf den Datenstapel gelegt. Nun folgt die Kompilation des Zweigprogramms. Danach muß der Datenstapel denselben Zustand haben, damit die Prüfung der Kennzahl 6 durch **ENDOF** positiv ausfällt. Von ENDOF wird außerdem der mit OF kompilierte bedingte Vorwärtssprung auf den nach ENDOF vorzunehmenden Wörterbucheintrag, das heißt die nächste Fallabfrage, gerichtet. Der Speicherplatz für die Adresse des im Falle der Übereinstimmung notwendigen unbedingten Sprungs ans Ende der Gesamtkonstruktion wird durch das in ENDOF stehende >**MARK** reserviert.
Das Wort **ENDCASE** löst die von den ENDOFs auf dem Datenstapel hinterlassenen Markierungsadressen (für die unbedingten Sprünge ans Ende der Konstruktion) auf. Durch Vergleich des Stapelzeigers mit dem in CSP abgelegten alten Wert wird geprüft, ob die richtige Anzahl von Adressen zugeordnet wurde.
Wir empfehlen Ihnen, die Kompilation und die Ausführung des folgenden, als Demonstration gedachten Wortes nachzuvollziehen:

: .X (n ===>)
CASE 0 OF ." Zweig 0 " ENDOF
1 OF ." Zweig 1 " ENDOF
2 OF ." Zweig 2 " ENDOF
." Zweig x" ENDCASE ;

Tafel 5 Kurzbeschreibung der Forthworte

Name	Stapeleffekt	Beschreibung
Wörterbuchverwaltung		
.	(16b ===>)	16b als neuen Eintrag ans Wörterbuch anhängen
C,	(8b ===>)	8b als neuen Eintrag ans Wörterbuch anhängen
ALLOT	(n ===>)	reserviert die nächsten n Byte im Wörterbuch
HERE	(===> addr)	addr ist die nächste freie Adresse am Wörterbuchende
IMMEDIATE	(===>)	erklärt das zuletzt definierte Wort zum Vorrangwort
Definitionsworte		
CREATE	(===>)	erzeugt den Kopf für eine Variable
DOES>	(===> pfa)	Beginn des für alle Elemente der Wortklasse gleichen Laufzeitcodes (Hochcode)
:CODE	(===>)	ähnlich wie DOES>, aber es folgt Assembler- oder Maschinencode
Kompilationsworte		
<MARK	(===> addr)	liefert die aktuelle Wörterbuchendeposition addr zum Stapel
<RESOLVE	(addr ===>)	benutzt die addr von <MARK zur Berechnung und Eintragung der Adresse eines Rückwärtssprunges
>MARK	(===> addr)	liefert die aktuelle Wörterbuchendeposition addr und reserviert Platz für eine Vorwärtssprungadresse
>RESOLVE	(addr ===>)	bildet aus der von >MARK gelieferten addr und der aktuellen Wörterbuchendeposition einen Vorwärtssprung und trägt diesen in den von >MARK reservierten Platz ein
COMPILE	(===>)	kompiliert in der Ausführungsphase das nachfolgende Wort
LITERAL	(Kompilation: n ===>) (Ausführung: ===> n)	kompiliert die auf dem Parameterstack liegende Zahl als Literal und liefert diese bei der Ausführung zum Stapel
STATE	(===> addr)	Variable, die während des Ausführungsmodus den Wert Null hat
[	(===>)	schaltet den Ausführungsmodus ein
]	(===>)	schaltet den Kompilationsmodus ein
[COMPILE]	(===>)	kompiliert das im Quelltext nächstfolgende Wort; auch dann, wenn es ein Vorrangwort ist

5.3.3. Zustandssteuerung

Grundsätzlich befindet sich das Forthsystem entweder im Modus *Kompilieren* oder im Modus *Ausführen.* Welcher Zustand gerade aktiv ist, kann der Systemvariablen **STATE** entnommen werden. Im Modus Ausführung hat die Variable STATE den Wert Null, ansonsten ist sie davon verschieden (implementationsabhängig). STATE ist für die Abfrage des Systemzustandes vorgesehen und darf vom Anwender nicht selbst verändert werden. Manchmal kommen in Programmen Rechenoperationen mit solchen Operanden vor, deren Werte schon zur Übersetzungszeit bekannt sind. In solchen Fällen wäre es günstig, die entsprechenden Operationen tatsächlich schon zur Übersetzungszeit zu erledigen, so daß im übersetzten Programm schon die fertigen Zwischenergebnisse verwendet werden. Die aufgewendete Rechenzeit spart man bei der wiederholten Abarbeitung des definierten Wortes mehrfach ein. Die Berechnungen erfordern den Zustand *Ausführung*, der während der Kompilation vorübergehend eingeschaltet werden müßte. Nach Berechnung des Wertes wird dann weiter kompiliert. Zur Umschaltung des Zustandes werden die beiden Worte

[ ]

benutzt. Die Notation läßt sich leicht merken, weil die in Klammern stehenden Ausdrücke als „Nebenrechnung" aufgefaßt werden können, die nicht kompiliert wird. Als Ergebnis der „Nebenrechnung" wird in der Regel ein Zahlenwert auf dem Datenstapel geliefert. Seine Kompilation kann mit dem Wort **LITERAL** erfolgen, das einen zur Compilationszeit auf dem Datenstapel liegenden Wert ins Wörterbuch einträgt. Beim Aufruf der Passage wird der Wert auf den Stapel zurückgebracht.

Beispielsweise kann für ein Objekt der Klasse **VEKTOR** mit dem Namen **VX** nach Abschnitt 5.2.1. der Zugriff auf die dritte Komponente anstatt durch

: zzz . . . VX 3 KOMPONENTE . . . ;

mit

: zzz . . . [VX 3 KOMPONENTE]
LITERAL . . . ;

vorgenommen werden. Der komplizierter aussehende zweite Quelltext stellt die günstigere Lösung dar. Statt der 4 ∗ 16 Bit Speicherplatz im Wörterbuch werden nur 2 ∗ 16 Bit benötigt. Außerdem wird während der Laufzeit des Wortes zzz die Passage schneller abgearbeitet, weil die in Klammern stehenden Operationen bereits während der Übersetzung in den Fadencode erledigt worden sind.
Die gewöhnlichen Zahlen werden übrigens während der Kompilation in derselben Weise behandelt. Zur Ablage einer Zahl im Kompilat werden deshalb in der Regel zweimal 16 Bit benötigt, einmal für LITERAL und einmal für die Zahl selbst. Das ist ein höherer Aufwand als für Konstanten, wo nur die Adresse (16 Bit) zur CFA der Konstanten ins Wörterbuch geschrieben wird. Damit gibt es einen weiteren Grund für die Verwendung von Konstanten.

wird fortgesetzt

Fortsetzung von Seite 334

MREQ einen Ressourcenzugriff einzuleiten, könnte folgendes ablaufen:

① MREQ führt den Test auf die Verfügbarkeit der Ressource aus. Dieser Test ist negativ. Es erfolgt keine Anmeldung.

② Zu diesem Zeitpunkt endet die Nutzung des Main-frame-RAM durch die Fremd-CPU.

③ Der Versuch der Anmeldung durch MREQ wird wiederholt. Der Test auf die Verfügbarkeit der Ressource fällt positiv aus. Daraufhin erfolgt die Anmeldung, indem an /mO der Signalpegel Low ausgegeben wird. Ein höherpriorisiertes System hatte aber gleichzeitig eine Anforderung ausgesandt, so daß nach Ablauf der Zeitkonstante am Eingang /mI noch High anliegt. Die Anforderung wurde also abgelehnt. Der Ausgang /mO wird wieder auf High gesetzt. In Bild 6 ist dieser Vorgang dargestellt.

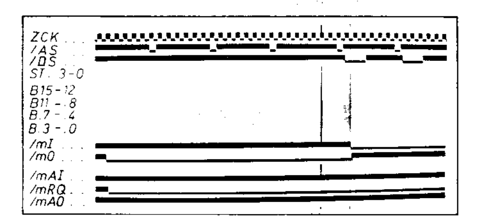


Bild 6 Ablehnung einer Multi-Micro-Anforderung

④ Der Transfer durch eine Fremd-CPU wird beendet.

⑤ Die CPU unternimmt erneut den Versuch einer Anmeldung. Der Test auf die Verfügbarkeit der Ressource fällt positiv aus. Nach der Laufzeit der Daisy-Chain wird /mI zu Low. Sobald die programmierte Zeitkonstante zu Null dekrementiert ist, wird der Signalpegel an /mI abgefragt. Da das Ergebnis positiv ist, bleibt der Ausgang /mO auf Low. Der Transfer mit dem Main-frame-RAM findet statt.

⑥ Auf die Beendigung des Transfers folgend wird durch den Befehl MRES der Ausgang /mO auf High gesetzt.

⑦ Erst nach der Laufzeit des Signals durch die Multi-Micro-Daisy-Chain wird der Eingang /mI auf High gesetzt. Dann ist die Ressource wieder frei verfügbar.

Eine erfolgreiche Anmeldung und Durchführung eines Ressourcenzugriffs ist vollständig in Bild 7 enthalten.

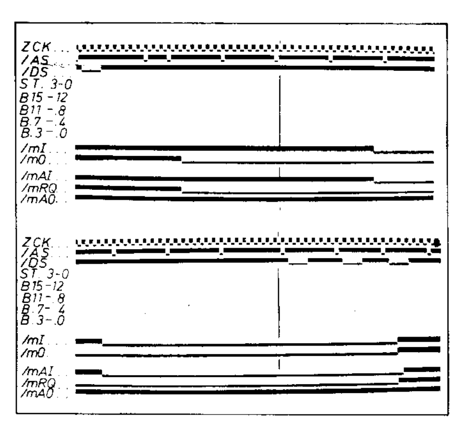


Bild 7 Erfolgreiche Anmeldung und Durchführung eines Ressourcenzugriffs

Beispielprogramm

Dieses Beispiel soll noch einmal den Umgang mit den Befehlen MREQ und MRES veranschaulichen. Zum besseren Verständnis des zu diesem Beispiel gehörenden Bildes 8 (Spreizung von Bildausschnitten des Bildes 7) wird der Programmzähler und der Befehlscode mit aufgeführt.

Literatur

/1/ Heuer, H: Parallelverarbeitende Rechnersysteme. Mikroprozessortechnik, Berlin 1 (1987) 3, S. 71

/2/ Leventhal, L. A.; Collins, C.; Osborne, A.: Z8000 Assembly Language Programming. Berkeley: OSBORNE/McGRAW-Hill 1980

/3/ Stuhlmüller, P.: 16-BIT-GENERATION, Z 8000, Aufbau und Anwendung. München: te-wi-Verlag 1980

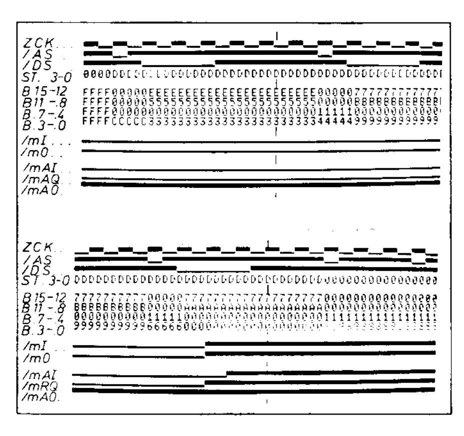


Bild 8 Auswertung der durch MREQ gesetzten Flags und Rücknahme der Ressourcenanforderung

```
        MULTI  MODULE
        CONSTANT
count:= %0007 ! Verzögerungskonstante
               für MREQ !

        $rel %0002
resfcw  WORD:=%4000
respc   WORD:=reset

        MAIN   PROCEDURE
        ENTRY
reset:  LD     R1,#count

        MREQ   R1
        JR     MI,available    ! S=1 ? !
        JR     Z,denied        ! Z=1 ? !
        HALT !Ressource nicht verfügbar!
denied: HALT !Anforderung abgelehnt!
available:
( Hier die Befehlsfolge für den
 Transfer mit der Ressource einbinden )
        MRES
        HALT

END     MAIN
END     MULTI
```

Bild 9 Beispielprogramm

✉ KONTAKT ☎

Wilhelm-Pieck-Universität Rostock, Sektion Technische Elektronik, WB Energiewandler, Albert-Einstein-Straße 2, Rostock, 2500; Tel. 45285

Hoher Anspruch an die Informatikausbildung

Jedem Studenten einer Hoch- oder Fachschule der DDR werden im Verlaufe seines Studiums im Rahmen der Informatikausbildung mindestens fachrichtungsbezogene Grundkenntnisse für die zielgerichtete Einführung und Nutzung computerintegrierter Systeme vermittelt. Damit wird dem zunehmenden Einsatz der Rechentechnik in allen Bereichen unserer Gesellschaft Rechnung getragen.

Darüber hinaus erhalten rund 10 bis 20 Prozent der Hochschulstudenten in den technischen, ökonomischen, agrarwissenschaftlichen, naturwissenschaftlichen und medizinischen Fachrichtungen notwendige Kenntnisse und Fähigkeiten zur Erarbeitung von Softwaresystemen für die Anwendung im jeweiligen Fachgebiet, das heißt eine entwicklerorientierte Informatikausbildung. Dies befähigt die Absolventen, auf ihrem späteren Arbeitsgebiet selbst nachnutzungsfähige spezifische Anwendungssoftware zu entwickeln. Die Ausbildung von Informatikspezialisten wurde in den vergangenen Jahren schrittweise erweitert. Das belegt die Zahl von jährlich mehr als 700 immatrikulierten Studenten. Parallel zum quantitativen Ausbau erhöhte sich auch das Niveau der Ausbildung. Dazu trugen die Gründung des Informatikzentrums des Ministeriums für Hoch- und Fachschulwesen an der Technischen Universität Dresden und neuer Hochschulsektionen, die Berufung einer Reihe namhafter Wissenschaftler als Hochschullehrer sowie die Erweiterung der rechentechnischen Ausstattung bei.

In Vorbereitung der wissenschaftlich-methodischen Informatikkonferenz, die im Februar diese Jahres in Dresden stattfand, zogen Wissenschaftler, Hoch- und Fachschullehrer sowie Studenten an den Universitäten, Hoch- und Fachschulen der DDR im Ergebnis zahlreicher Beratungen Bilanz über die bisher auf dem Gebiet der Informatik in Lehre und Forschung erreichten Ergebnisse; sie verständigten sich über neue Aufgabenstellungen. Gegenstand ihres offenen und konstruktiven Meinungsaustausches waren vor allem die vom Wissenschaftlichen Beirat für Informatik ausgearbeiteten Thesen über die gesellschaftliche Bedeutung der Informatik und die Entwicklung dieser Fachdisziplin an den Universitäten und Hochschulen. Erörtert wurden zudem die Aus- und Weiterbildung von Spezialisten auf diesem Gebiet sowie pädagogische Fragen, die sich aus dem Einsatz von Computern in der Aus- und Weiterbildung ergeben.

In den gründlichen Beratungen zu diesen Thesen wurde deutlich, daß sich der im Hoch- und Fachschulwesen eingeschlagene Weg der Informatikausbildung zunehmend bewährt. Die Studenten erhalten entsprechend dem Studienziel entweder eine nutzerorientierte oder eine umfassendere entwicklerorientierte Informatikausbildung. Darüber hinaus werden Kader für die Weiterentwicklung der Wissenschaftsdisziplin Informatik, für die Entwicklung von Basissoftwaresystemen vor allem in den Kombinaten Robotron und Datenverarbeitung sowie als Leiter von Rechenzentren ausgebildet. Mit der schrittweisen Realisierung des Konzepts der Informatikausbildung, das zugleich die Basis für entsprechende Aktivitäten in der Weiterbildung ist, sind hohe Anforderungen an die Studenten und den Lehrkörper gestellt. Für die Studenten ergeben sich daraus nicht nur neue Qualitätsansprüche an ihr Grundlagen- und Fachwissen wie zum Beispiel in der Mathematik. Sie haben sich zugleich auf zum Teil bisher ungewohnte Arbeitsweisen bei der Rechnernutzung in den einzelnen Lehrgebieten einzustellen. Besonders die Methoden der Modellierung und Simulation von Sachverhalten und Prozessen in Natur, Technik und Gesellschaft sind zunehmend Gegenstand der Ausbildung.

Die Hochschullehrer werden sowohl fachlich als auch methodisch-pädagogisch stärker gefordert und bedürfen einer zunehmenden Qualifikation. Hierin liegen noch wesentliche Reserven für die weitere Steigerung des Ausbildungsniveaus. Darauf verwies auch der Minister für Hoch- und Fachschulwesen, Prof. Dr. h. c. Hans-Joachim Böhme, auf der bereits genannten Konferenz. Er bekräftigte die Notwendigkeit, die Qualität der entwicklerorientierten Informatikausbildung zu steigern und dazu insbesondere die Qualifikation der Hochschullehrer zu erhöhen. Der Minister unterstützte auch den Vorschlag des Wissenschaftlichen Beirates für Informatik, durch Neugestaltung der Fachrichtungsstruktur in der Grundstudienrichtung Informatik das Niveau der Ausbildung von Spezialisten zu verbessern und die Disponibilität der Absolventen zu fördern. Im Interesse der Weiterbildung bereits in der Praxis tätiger Kader ist es erforderlich, die Arbeiten an einem Studienplan für das postgraduale Studium „Angewandte Informatik – CAD/CAM“ mit einem Fachschulabschluß für Softwareentwicklung zu beschleunigen.

Dr. U. Dietzsch

Portabler Bildschirmzugriff in Unix

Eckhard Einert, Ulrike Luthe
Technische Universität
Karl-Marx-Stadt

Sicherlich ist es nicht notwendig, an dieser Stelle noch große Worte über das Betriebssystem Unix zu verlieren. Weltweit wird dieses System auf verschiedenen Rechnern der unterschiedlichsten Architekturen und Größen betrieben. Zum gegenwärtigen Zeitpunkt sind vielfältige Bestrebungen auf den unterschiedlichsten Ebenen im Gange, um Unix im allgemeinsten Sinne zu standardisieren /1/. Die Hauptzielrichtung all dieser Standardisierungsbestrebungen ist die Definition einer Anwendungsschnittstelle, die vom Basisbetriebssystem bereitgestellt wird und das Fundament für eine einheitliche Entwicklungsumgebung bildet. Damit kann die Portabilität der auf diesen standardisierten Unix-Systemen geschriebenen Anwendungen wesentlich erhöht werden.
Das Terminal stellt die unmittelbare Schnittstelle zum Nutzer dar. Darum ist im Rahmen der oben genannten Standardisierungsbestrebungen auch die Vereinbarung eines allgemeinen, von der Hardware und der Anschlußmethode unabhängigen Terminalinterfaces für alle Anwendungen notwendig. Dieses Interface soll das Betreiben von zeichen- und blockorientiert arbeitenden Terminals gestatten, unabhängig davon, wie diese mit dem System verbunden sind (Lokalverbindung, über LAN, als PAD in X.25 usw.).

Als ein erster Schritt in diese Richtung ist die Definition der Curses-Bibliotheksroutinen für eine allgemeine Terminalbehandlung, erstmals in den BSD-Systemen anzutreffen, durch X/OPEN /2/ zu betrachten. Die Curses-Bibliothek stellt dem Nutzer eine komfortable Methode für einen *optimalen geräteunabhängigen Bildschirmzugriff* zur Verfügung. Die Funktionen sind auf lokal verbundene asynchrone alphanumerische Terminals ausgerichtet. Anwendungen, die dieses Interface beachten, sind für solche Terminals portabel. Probleme können beim Betreiben von synchronen, asynchronen Nichtstandard- bzw. Netzwerkterminals auftreten. Solche Terminals sind meist in Großrechnerumgebungen zu finden (z. B. EC 792x an den EC 10xx unter VMX /3/). Sie kommunizieren mit der Zentraleinheit im Blockmodus. Das heißt, der Nutzer gibt Zeichen am Terminal ein und muß erst eine spezielle Taste betätigen (ENTER, PA, PF), um die Übertragung der Zeichen in das System zu erreichen. Diese Eigenschaft kann zu Problemen führen, wenn man in einer Anwendung von der Einzelzeicheneingabe Gebrauch machen möchte. Des weiteren kann das Sperren des Echos Schwierigkeiten bereiten, da das Echo im allgemeinen unmittelbar durch das Terminal ausgeführt wird und die Eingabe direkt an der Stelle erscheint, an der sich der Kursor befindet. Dessen Position muß jedoch nicht mit der Position des Kursors im Fenster übereinstimmen. Die Ausgaben müssen sich auf diesen Bildschirmen ohne Einschränkungen (außer eventuellem Geschwindigkeitsverlust) abbilden lassen.
Das Curses-Paket wird in unterschiedlichen Ausbaustufen von den verschiedensten Systemen bereitgestellt, wobei teilweise noch erhebliche Abweichungen zum Standardvorschlag als auch zur Dokumentation zu bemerken sind /4/ /5/ /6/.
Die Terminalunabhängigkeit des Pakets basiert auf einer verallgemeinerten Terminalbeschreibung in einem TERMCAP-File /7/. In diesem speziellen Unix-File werden die konkreten Zeichenfolgen beschrieben, die notwendig sind, um eine bestimmte Funktion auf einem gegebenen Terminal auszuführen. Der Nutzer gibt über den Wert der Shell-Variablen TERM seinen Terminalnamen an. Dieser Name bestimmt, welche Beschreibung im TERMCAP-File (/etc/termcap) für das Ansprechen der Terminals Verwendung findet.

Übersicht

Die Grundlage für die weiteren Betrachtungen stellt die X/OPEN Terminal Interfaces Definition /2/ dar, da alle Curses-Pakete in Zukunft an dieser Definition gemessen werden müssen.
Die Bibliotheksfunktionen greifen auf den Terminalbildschirm zu, indem sie mit den internen Datenstrukturen *Bildschirm* und *Fenster* arbeiten. In der Originaldokumentation werden dafür die Begriffe *Screen* und *Window* verwendet. Die Struktur *Bildschirm* enthält das Aussehen des gesamten zukünftigen Bildschirms, und ein Fenster ist die Repräsentation eines Teils des zukünftigen Bildschirms. In einem Bildschirm sind ein oder mehrere *Fenster* abbildbar. Ein *Fenster* kann aus einem einzelnen Zeichen bestehen, aber auch so groß wie der gesamte Bildschirm sein, oder aber nur partiell auf dem Schirm abgebildet werden. Des weiteren besteht die Möglichkeit, *Fenster* zu erzeugen, die größer als der aktuelle Bildschirm sind, sogenannte Pads.
Bevor Bildschirme oder *Fenster* benutzbar sind, müssen sie unter Verwendung der Funktionen *newwin()* oder *subwin()* erzeugt werden. Diese Funktionen definieren die Größe des *Bildschirms* oder des *Fensters* und die Position der linken oberen Ecke wie folgt:

newwin (<Zeilenanzahl>, <Spaltenanzahl>, <oberste Zeile>, <äußerste linke Spalte>)

Über diese Strukturen ist ein Koordinatensystem gezogen, welches ebenfalls in Zeilen und Spalten eingeteilt ist. So hat das Zeichen in der linken oberen Ecke des *Bildschirms* die Koordinaten (0,0). Ein typischer *Bildschirm* hat 80 Spalten und 24 Zeilen. Seine linke obere Ecke entspricht der linken oberen Ecke des (physischen) Terminalbildschirms. Ein *Fenster* dagegen darf von beliebiger Größe sein, die linke obere Ecke eines *Fensters* kann jeder beliebigen Position auf dem Bildschirm entsprechen. Durch die Funktion *initscr()*, die ein Programm für die Bildschirmverarbeitung initialisiert, wird ein Standardschirm *stdscr* (Standard Screen) bereitgestellt, der dem gesamten Bildschirm entspricht und nicht erst erzeugt werden muß.
Alle weiteren genannten Funktionen sind auf einen Bildschirm oder aber auch auf ein Fenster bezogen (ausgedrückt durch das Präfix **w** im Funktionsnamen). Um Zeichen auf dem Terminalbildschirm anzuzeigen, müssen diese zuvor durch das Programm in die Strukturen *Bildschirm* oder *Fenster* unter Verwendung solcher Funktionen wie *addch()* bzw. *waddch()* geschrieben werden, wobei eine Positionierung mit den Funktionen *move ()* bzw. *wmove()* erfolgen kann. Sollen nun diese Zeichen auch auf dem physischen Bildschirm erscheinen, so kann das Programm das Aussehen des Terminalbildschirms unter Verwendung der Funktionen *refresh()* oder *wrefresh()* aktualisieren. Das Paket bestimmt dabei, welche Veränderungen welche Auswirkungen auf den physischen Schirm haben. Der Terminalbildschirminhalt wird danach unter Verwendung der für den konkreten Terminaltyp optimierten Operationen auf das geforderte Aussehen gebracht. Da sich die Aktualisierung des Terminalbildschirms solange verzögert, bis ein Programm die Funktionen *refresh()* oder *wrefresh()* ruft, besteht die Möglichkeit, verschiedene Fenster zu verwalten, die unterschiedliche Zeichen für die gleiche Terminalbildschirmposition enthalten. Das Programm kann vor der Aktualisierung auswählen, welches Fenster zur Anzeige zu bringen ist.
Mehrere *Fenster* sind auf dem Bildschirm unter Verwendung der Funktionen *overlay()* und *overwrite()* miteinander anordenbar.
Es existieren aber auch Routinen, die komfortablere Ausgabemöglichkeiten ähnlich dem bekannten *printf()* zur Verfügung stellen.
Das Interface gestattet das Löschen ganzer *Fenster* und auch das Setzen von Bildschirmattributen, wie invers, unterstrichen oder blinkend, für einzelne Zeichen im Fenster. Für große Anwendungen, die mehrere Terminals gleichzeitig betreiben möchten, besteht die Möglichkeit, mit der Funktion *newterm()* zusätzliche Terminals einzubeziehen.
Im Interface sind ebenso Zeicheneingabefunktionen definiert, wobei die Manipulation der eingegebenen Zeichen erlaubt oder gesperrt werden kann (Echo, Einzelzeicheneingabe, Signalverarbeitung, Abbildung von \r auf \n, Bildschirmscrolling usw.).
Zur Definition der durch das *Curses*-Paket genutzten Typen und Variablen wird das Headerfile **curses.h** zum Einschluß in die Nutzerquellprogramme bereitgestellt.

Funktionsgruppen

An dieser Stelle sei noch einmal darauf hingewiesen, daß die meisten Funktionsgruppen paarweise existieren, für den gesamten *Bildschirm* und für ein einzelnes *Fenster*. Das gewünschte *Fenster* wird dabei als erstes Argument der Funktion angegeben. Für eine kompaktere Darstellung seien im weiteren nur die Funktionen für den Zugriff auf den *Bildschirm stdscr* aufgeführt. Die meisten Funktionen sind aber unter Beachtung der oben genannten Konventionen auf beliebige Fenster anwendbar. Die Tafeln 1 bis 3 enthalten eine Übersicht über die wichtigsten Funktionen des *Curses*-Pakets. Die Funktionen ohne Äquivalent mit dem w-Präfix wurden durch ein # gekennzeichnet.

Ausgabefunktionen

Die Basisfunktionen, um einen Fenster-Inhalt zu ändern, sind **addch()** und **move()**. addch() gibt ein Zeichen an der aktuellen (x,y) Cursorposition aus und setzt den Cursor weiter. Mit move() kann die aktuelle (x,y) Cursorposition

```
#include <curses.h>

/* Dialogfenster */
#define YDIA     10
#define XDIA     27
#define YPOSDIA  6
#define XPOSDIA  0
#define BOXDIA  '*'

/* Ergebnisfenster */
#define YERG     16
#define XERG     35
#define YPOSERG  6
#define XPOSERG 30
#define BOXERG  '='

/* Fehlermitteilungsfenster */
#define YFEH      6
#define XFEH     59
#define YPOSFEH 18
#define XPOSFEH  2
#define BOXFEH  '.'

main()
{
WINDOW *dialog,*ergebnis,*fehler;
char zeichen;

initscr();

/* Einrichten der Fenster */
dialog=newwin(YDIA,XDIA,YPOSDIA,XPOSDIA);
ergebnis=newwin(YERG,XERG,YPOSERG,XPOSERG);
fehler=newwin(YFEH,XFEH,YPOSFEH,XPOSFEH);
/* Ausgabe Führungstext */
mvaddstr(2,13,"Beispiel zur Nutzung mehrerer Fenster");
refresh();

/* Anfrage */
box(dialog,BOXDIA,BOXDIA);
mvwaddstr(dialog,3,3,"Geben Sie eine Ziffer");
mvwaddstr(dialog,4,3,"ein:");
for(;;) {
  wmove(dialog,6,10);
  wrefresh(dialog);
  zeichen=wgetch(dialog);
  box(ergebnis,BOXERG,BOXERG);
  mvwaddstr(ergebnis,2,7,"Ergebnis:");
  wclear(fehler);
  wrefresh(fehler);

  if ( zeichen < '0' || zeichen > '9' ) {
    /* Fehlerbehandlung */
    box(fehler,BOXFEH,BOXFEH);
    mvwprintw(fehler,2,8," Eingegebenes Zeichen %c",zeichen);
    mvwprintw(fehler,2,31," ist keine Ziffer ! ");
    mvwprintw(fehler,4,13," Wiederholen Sie bitte ! ");
    overwrite(ergebnis,stdscr);
    overwrite(fehler,stdscr);
    mvwaddch(dialog,6,10,' ');
    refresh();
  } else {
    /* Behandlung gültiger Eingabe */
    mvwprintw(ergebnis,5,5,"Das eingegebene Zeichen");
    mvwprintw(ergebnis,7,15,"%c",zeichen);
    mvwprintw(ergebnis,9,5,"ist eine gueltige Ziffer");
    mvwprintw(ergebnis,10,5,"und kann weiterverarbeitet");
    mvwprintw(ergebnis,11,5,"werden...");
    touchwin(ergebnis);
    wrefresh(ergebnis);
    break;
  }
}

/* Hier eventuelle Weiterverarbeitung der Eingabe ... */

mvcur(0,COLS-1,LINES-1,0);
endwin();
}
```

◀ Bild 1

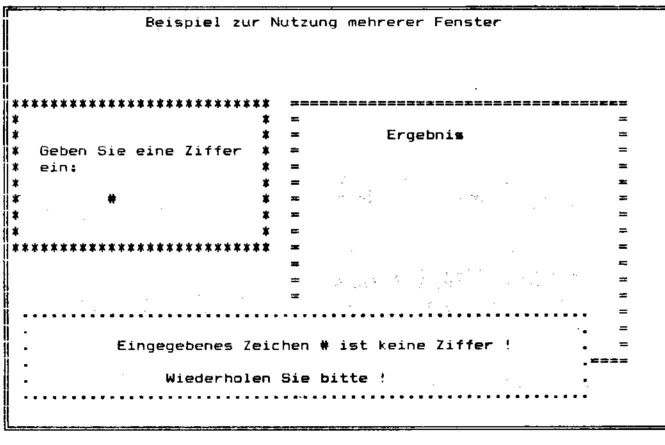
Tafel 1 Curses-Ausgabefunktionen

Funktionsname	Beschreibung
addch()	Ausgabe Zeichen auf Fenster
mvaddch()	wie addch(), mit Cursorpositionierung
addstr()	Ausgabe einer Zeichenkette auf Fenster
mvaddstr()	wie addstr(), mit Cursorpositionierung
beep()	Ausgabe eines akustischen Signals
box()	Umrahmen Fenster
clear()	Löschen Fensterinhalt
clrtobot()	Löschen bis Ende des Fensters
clrtoeol()	Löschen bis Zeilenende
delch()	Löschen Zeichen in Fenster
mvdelch()	wie delch(), mit Cursorpositionierung
deleteln()	Lösche Zeile von Fenster
insch()	Einfügen eines Zeichens
mvinsch()	wie insch(), mit Cursorpositionierung
insertln()	Einfügen einer Zeile
move()	Cursorpositionierung im Fenster
printw()	formatierte Ausgabe auf Fenster
mvprintw()	wie printw(), mit Cursorpositionierung

Tafel 3 Curses-Steuerfunktionen

Funktionsname	Beschreibung
# cbreak()	Setzen Löschen CBREAK-Modus
# delwin()	Fenster beseitigen
# echo()	Erlauben/Sperren Terminalecho
# endwin()	Wiederherstellen Anfangsterminal-umgebung
# getyx()	Holen Cursorposition
# initscr()	Initialisieren Curses-Umgebung
# mvwin()	Fenster bewegen
# newpad()	Erzeugen eines neuen pads
# newterm()	Eröffnen eines neuen Terminals
# overlay()	Überlagern von Fenstern (außer Space)
# overwrite()	wie overlay(), mit Spaceüberlagerung
# prefresh()	Anzeige Teil eines pads auf Terminal-bildschirm
# raw()	Setzen/Löschen RAW-Modus
refresh()	Anzeige Fenster auf Terminalbildschirm
# scroll()	Rollen Fensterinhalt
# set_term()	Terminalwechsel
# subwin()	Erzeugen eines Unterfensters
# touchwin()	ganzes Fenster als aktualisiert kenn-zeichnen

Bild 2 ▼



Tafel 2 Curses-Eingabefunktionen

Funktionsname	Beschreibung
getch()	Lesen Zeichen
mvgetch()	wie getch(), mit Cursorpositionierung
getstr()	Lesen Zeichenkette
mvgetstr()	wie getstr(), mit Cursorpositionierung
inch()	Lesen Zeichen von Fenster
mvinch()	wie inch(), mit Cursorpositionierung
scanw()	formatiertes Lesen
mvscanw()	wie scanw(), mit Cursorpositionierung

verändert werden. Die Wirkung dieser beiden Routinen wird in der Funktion mvaddch() zusammengefaßt. Komplexe Ausgabefunktionen wie addstr() und printw() (ähnlich printf() (3)) werden mit addch erreicht. Bildschirmattribute können mit den Zeichen verbunden werden. Bei gesetztem Scroll-Attribut wird bei Erreichen des Fensterendes ein automatisches Scrolling innerhalb des Fensters ausgeführt.

Das Fenster kann mit einem Rahmen aus beliebigen Zeichen versehen werden, wobei zu beachten ist, daß dieser Rahmen mit in die dem Fenster gehörigen Zeichenpositionen eingeht.

Verschiedenste Löschfunktionen werden bereitgestellt, vom Löschen eines einzelnen Zeichens bis zum Löschen des gesamten Fensters.

Um Zeichen an einer bestimmten Position einzufügen, unter Weiterrücken der folgenden, existieren die Funktionen **insch()**, **insertln()**, **mvinsch()**. Der Inhalt eines Fensters kann in ein anderes unter Verwendung der Funktionen **overlay()** bzw. **overwrite()** kopiert werden.

Eingabefunktionen

Unix unterscheidet drei Eingabemodi:

CBREAK-Modus

Das Zeichen ist sofort nach der Eingabe für die Anwendung verfügbar; die Erase-Kill-Zeichenverarbeitung wird nicht ausgeführt, und die Wirkung der Interrupt- und Flußsteuerzeichen bleibt erhalten.

RAW-Modus

wie CBREAK-Modus, aber Interrupt- und Flußsteuerzeichen werden nicht interpretiert durchgereicht, so daß sie kein Signal generieren.

COOKED-Modus

Zeichen werden bis zur Eingabe von *Newline* oder *Carriage Return* gepuffert.

Durch das *Curses*-Paket werden alle diese Eingabemodi unterstützt. Das Echo eines Zeichens ist möglich, kann aber auch gesperrt werden. Mit *getch()* bzw. *mvgetch()* kann ein Zeichen vom Terminal gelesen und in (falls erforderlich) ein Fenster ausgegeben werden. Im *Nodelay*-Modus kehrt diese Funktion sofort zurück und wartet nicht erst bis ein Zeichen eingegeben wurde. Mit *getstr()* bzw. *mvgetstr()* und *scan()* bzw. *mvscanw()* (ähnlich *scanf()*) sind ganze Zeichenketten lesbar. Zeichen können auch direkt aus einem *Fenster* mit *inch()* bzw. *mvinch()* gelesen werden.

Steuerfunktionen

Mit speziellen Funktionen ist ein Programm in der Lage, bestimmte Eigenschaften des eingesetzten Terminals abzufragen und zu steuern. So ist es beispielsweise möglich, die aktuelle Cursorposition zu ermitteln oder die sichtbare Cursorbewegung zu unterdrücken. Fenster und Unterfenster können erzeugt, bewegt und gelöscht werden (newwin(), subwin(), mvwin(), delwin()). Insbesondere durch die Funktion subwindow() kann das leidige Problem des Rahmens umgangen werden.

Die Bildschirmarbeit mit dem Curses-Paket ist mit der Funktion initscr() zu initialisieren, wobei der für die Arbeit notwendige Speicher bereitgestellt und das Terminal in den geforderten Modus versetzt wird. Mit endwin() erfolgt das Verlassen der Bildschirmverarbeitung, das Terminal wird in seinen vorherigen Zustand zurückgesetzt und der Speicher freigegeben.

Dieser Funktionsaufruf **muß** unbedingt vor dem Verlassen des Programms erfolgen.

Beispiel

In einem einfachen Programm soll nun die Verwendung einiger Funktionen des Paketes illustriert werden (Bild 1). Für die Arbeit werden drei Fenster (dialog, ergebnis und fehler) initialisiert, deren Inhalt später in Abhängigkeit von bestimmten Ereignissen auf dem Bildschirm sichtbar gemacht werden soll. Durch den Aufruf von *mvaddstr()* erfolgt in den oberen Teil des Bildschirms die Ausgabe einer Überschrift. Das Fenster dialog wird umrahmt und in die dritte Zeile gibt *mvaddstr()* die Aufforderung zur Eingabe einer Ziffer aus. Erst danach wird dieses Fenster durch *wrefresh()* auf den Bildschirm gebracht (siehe Bild 2). Die Eingabe eines Zeichens beendet die Funktion *wgetch()*. Dabei wird ausgenutzt, daß die Funktion wgetch() bei erlaubtem Echo automatisch refresh() aufruft und das eingegebene Zeichen auf der vorher durch die Funktion wmove() gesetzten Cursorposition ausgibt. Nach der Prüfung, ob das eingegebene Zeichen eine Ziffer ist, erscheint in dem Fenster *ergebnis* eine Mitteilung über die Möglichkeit einer Weiterverarbeitung und das Programm endet bzw. es wird in einem Fenster *fehler* zu einer erneuten Eingabe aufgefordert.
Anhand dieser kurzen Vorstellung des *Curses*-Pakets läßt sich leicht erkennen, daß sich ganz einfach Prinzipien der Arbeit mit Fenstern unter Zuhilfenahme dieses Pakets nutzen lassen. Bereits nach kurzer Einarbeitung können auch anspruchsvolle Zielstellungen erreicht werden. Allerdings ist für die in /8/ vorgestellte komfortable Nutzerschnittstelle für Unix diese Funktionen jedoch unzureichend.
Auf dem Gebiet der eigentlichen Fenster für Unix zeichnen sich allerdings international ebenfalls schon Standardisierungsbestrebungen ab /9/.

Literatur
/1/ Clauss, M.; Fischer, G.: Zur Standardisierung von UNIX und C, EDV-Aspekte, (1987) 4, S. 11
/2/ X/OPEN Portability Guide, System V Spezifikation, Volume 3, Supplementary Definitions, Elsevier Science Publishers B. V., Amsterdam 1987
/3/ Fischer, G.; Clauss, M.: VMX – Virtual Machine UNIX – UNIX für ESER II. EDV-Aspekte (1987) 4, S. 13
/4/ Kenneth, C. R.; Arnold, C.: Screen Updating and Cursor Movment Optimization: a Library Package. University of California, Berkeley, California.
/5/ Kombinat EAW: WEGA-Software, Dienstprogramme (Band-B), Teil 3, SCREEN/CURSES – Bibliothek zur Bildschirmarbeit. Ausgabe 12/87
/6/ Microsoft Corporation: The XENIX System V, Development System, C Library Guide. 1987
/7/ Microsoft Corporation: The XENIX System V, Operating System, User's Reference 1987
/8/ Einert, E.: Fenstersysteme für Einzelplatz-Arbeitsstationen. EDV-Aspekte (1987) 4, S. 57
/9/ Rumpf, C.: X: Die Chance zum Standard für Windows, unix/mail 6 (1988) 1, S.19

KONTAKT
Technische Universität Karl-Marx-Stadt, PSF 964, Karl-Marx-Stadt, 9010; Tel. 668390

MES

Ein dialogfreundliches Echtzeitbetriebssystem

Dr. Frank Bonitz, Jörg Neunast, Arno Rockmann, Jan Rudorfer
Technische Hochschule Ilmenau

MES ist ein konfigurierbares Echtzeitbetriebssystem auf der Grundlage des Mikroprozessorsystems U 880. Aufbauend auf einem relativ kleinen Kern kann es aufgabenspezifisch erweitert werden. Mittels der im Kern realisierten Funktionen (Bild 1) können bis zu 255 Tasks prioritätsgerecht und unter Echtzeit abgearbeitet werden. Die kleinste Zykluszeit einer Task beträgt 1 tick (generierbar, min 10 ms), die maximale ist praktisch unbegrenzt.
MES realisiert eine dynamische Taskverwaltung. Tasks werden entweder durch Anfangsinitialisierung oder während des Laufs durch andere Tasks erzeugt. Damit können Anzahl und Auwahl der jeweils existierenden Tasks vom aktuellen Zustand des Systems oder seiner Umgebung beeinflußt werden (On-line-Rekonfiguration). Alle Systemdienste sind durch Interruptserviceroutinen nutzbar, was dem Anwender weitreichende Möglichkeiten der Reaktion auf externe Ereignisse einräumt. Zur Synchronisation des Zugriffs auf globale Ressourcen (Betriebsmittel) werden 16 Semaphore verwaltet. Die Semaphore können bei Bedarf direkt zur Tasksynchronisation genutzt werden.
Im Systemkern von MES sind Überwachungsroutinen integriert, die sowohl den ordnungsgemäßen Lauf von Tasks als auch systeminterne Abläufe überwachen. Beim Auftreten eines Fehlers wird je nach dessen Bedeutung eine selbständige Behebung eingeleitet und eine Meldung bereitgestellt oder das System stillgelegt. Für die Verwaltung der Meldungen, ihre Anzeige als Systemalarm und die Organisation der Quittierung steht die Systemkomponente *Fehlerverwaltung* zur Verfügung. Sie ist in Form eines Systemrufes auch für die Meldung von Prozeßalarmen und anderen Fehlerquellen nutzbar. Der Einsatz von MES in Leitrechnern wird durch einen leistungsfähigen Echtzeitmonitor unterstützt. Er besteht aus Treibern für Tastatureingaben und Displayausgaben sowie aus komplexen Monitorroutinen. Die wichtigsten dieser Monitorroutinen sind:

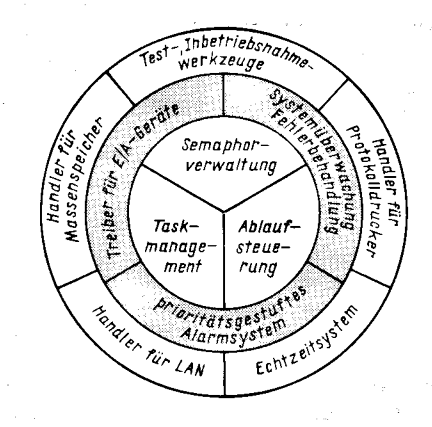
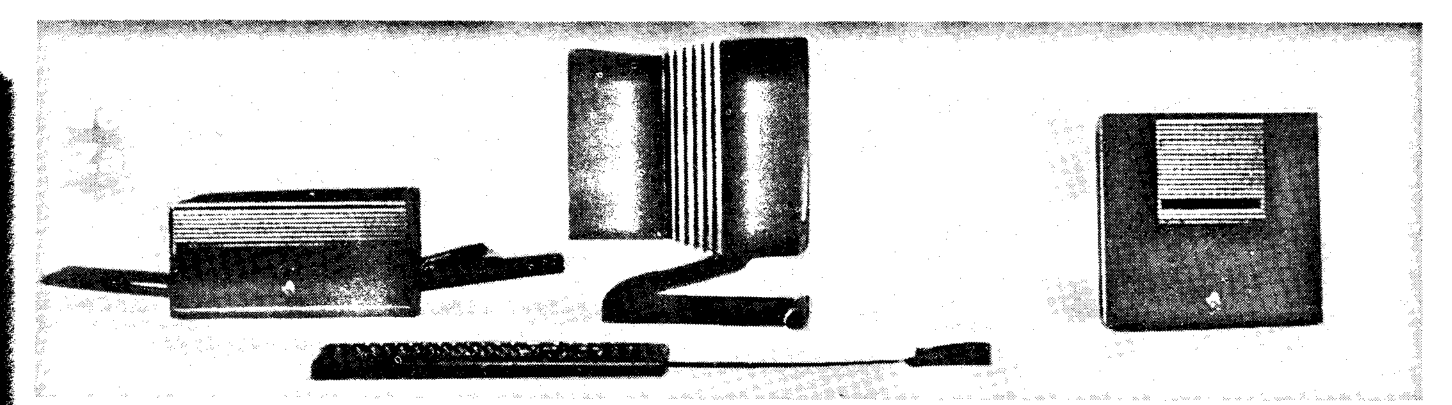


Bild 1 Komponenten des Echtzeitbetriebssystems MES

- Zeichenketteneingabe (Echo auf Display)
- Zahleneingabe (Echo auf Display), Bereitstellung als Hexadezimalzahl in Registern
- Alternativentscheidung (Echo auf Display)
- Zahlenausgaben (1 oder 2 Byte, Hexa- oder Dezimaldarstellung)
- Zeichenkettenausgabe
- Displayausschnitte löschen
- Ein- und Ausgabe eines Zeichens
- Schlüsselwortabfrage zur Prüfung der Bedienberechtigung für geschützte Programme.

Aufrufende Tasks, die auf Tastatureingaben warten, werden für die Dauer der Monitoroperation vom Tastaturtreiber suspendiert. In Abhängigkeit vom gewählten Eingabemodus (z. B. Zahleneingabe) wird nach Gültigkeitserklärung (ENTER) eine syntaktische Prüfung durchgeführt (Art und Anzahl der eingegebenen Zeichen). Bei Fehlererkennung leitet der Treiber einen einheitlichen Melde- und Quittierungsmechanismus ein. Die Schnittstelle für diesen Mechanismus ist auch dem Anwender zugänglich (z. B. bei semantischen Prüfungen). Bei Abschluß der Eingabeoperation wird der Wiederstart der suspendierten Task vom Tastaturtreiber veranlaßt. Das Display ist in mehrere physische und funktionelle Bereiche unterteilt, die als Betriebsmittel vom Betriebssystem verwaltet werden (Kopfzeile, Bildteil, Dialogteil, Bereiche für Vorrangmeldungen). Damit ist gesichert, daß unabhängig von Dialog- und Anzeigehandlungen eines Bedieners verschiedene Kategorien von Alarmen und Systemfehlern quasiparallel angezeigt und über die zugeordneten Sondertasten quittiert werden können.
Die Einbindung und Inbetriebnahme von Anwenderprogrammen unterstützt MES durch einen Echtzeit-Debugger und eine komfortable Handbedienung der Taskverwaltung und des Massenspeichers. Zur Programmentwicklung sind alle unter SCP verfügbaren Softwarewerkzeuge nutzbar. Für die interaktive Arbeit steht eine MES-Version unter SCP zur Verfügung.
Durch MES werden interruptgesteuerte Treiber und Handler für Drucker (IFSS) und Massenspeicher (MFS 1.6, FS 8" oder Magnetband) mit SCP-kompatiblem Aufzeichnungsformat bereitgestellt. Der Monitor kann für alphanumerische Anschlußsteuerungen mit 80 × 24 Zeichen oder die graphische Anschlußsteuerung VIS2A generiert werden, als Tastatur können die Typen K 7672.xx oder K 7643.xx eingebunden werden. Der Systemkern enthält für unterschiedliche Speicherkarten eine Speicherverwaltung bis 250 KByte.
MES ist sowohl auf Systemen mit ROM als auch mit RAM lauffähig und wurde auf OEM-Rechnern und Finalprodukten (K 89xx, IPC 8) implementiert. Auf der Grundlage der MES-Monitorroutinen und -Systemrufe steht ein umfangreiches Spektrum konfigurierbarer Anwenderprogramme für den Aufgabenbereich eines Wartenrechners zur Verfügung. Das Echtzeitbetriebssystem MES wird in Industrieanwendungen für Prozeßrechner- und Leitrechnerkonfigurationen eingesetzt.

KONTAKT
Technische Hochschule Ilmenau, Sektion Technische und Biomedizinische Kybernetik, Wissenschaftsbereich Automatische Steuerungen, Ilmenau, 6300; Tel. 2485



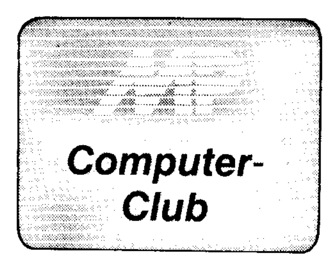
Nachdem die Euphorie um den Zauberwürfel des ungarischen Professors Rubik (Rubik's Cube) schon lange abgeklungen ist, macht in der Computerbranche jetzt ein anderer „Zauberwürfel" von sich reden. Zumindest beabsichtigt dies sein Schöpfer Steven Jobs. 1977 hatte der damals 21jährige mit Steve Wozniak die inzwischen legendäre „Garagenfirma" Apple gegründet und mit den Apple-PCs wesentlich der Idee vom persönlichen Computer zum Durchbruch verholfen (siehe auch unseren Beitrag „Der Apple Macintosh" im vorigen Heft). Nach Differenzen mit dem Management des expandierenden Unternehmens verließ Jobs dieses jedoch 1985 und gründete mit einigen Spitzenleuten, die er gleich mitnahm, die nächste Firma – beziehungsreich NeXT Inc. genannt. Mit dieser Basis beabsichtigte er, wiederum Maßstäbe zu setzen und der Welt den „PC der 90er Jahre" zu präsentieren. Zwar dauerte die Entwicklung wesentlich länger als angekündigt, aber am 12. Oktober 1988 war es soweit: Ähnlich wie beim Macintosh stellte er mit von den US-Amerikanern offensichtlich so geschätztem Brimborium in der Davis Symphony Hall von San Franzisko seine neue Schöpfung vor. Sowohl äußerlich als auch im Inneren besitzt der NeXT-Computer jedoch durchaus interessante Details, die ihn von den bisherigen Mikrorechnern abheben und eine größere Beachtung rechtfertigen. Da wäre zunächst die Bauform: Der eigentliche Computer steckt in einem von der Firma Frog Design gestalteten mattschwarzen Würfel von etwa 30 cm Kantenlänge, was ihm den Beinamen „The Cube" (Der Würfel) einbrachte. An diesem Würfel befinden sich keinerlei Bedien- oder Anzeigeelemente, sondern lediglich das Netzkabel zur Versorgung des gesamten Systems, mehrere Steckerleisten sowie ein drei Meter langes Kabel zum 17-Zoll-Monochrommonitor. An den Monitor ist die Tastatur, an der Tastatur wiederum die Maus angeschlossen. Der Würfel kann somit auch relativ weit entfernt vom Arbeitsplatz aufgestellt werden. Wie das Bild zeigt, sind alle Elemente des Computersystems – Zentraleinheit, Monitor, Tastatur und (wahlweise) Laserdrucker – in einheitlichem Design gestaltet. Aber auch vom inneren Aufbau her weicht der NeXT-Computer von herkömmlichen PCs ab. So gibt es nicht die bekannte Mutterplatine, auf der Erweiterungsleiterkarten eingesteckt werden können; vielmehr belegt die $33 \times 33\,cm^2$ große Hauptplatine selbst – wie zum Beispiel beim Macintosh – einen der vier 32-Bit-Steckplätze im NuBus-Standard. Dieses sogenannte Backbonekonzept wird bei eventuellen Erweiterungen als flexibler im Vergleich zur Mutterplatinenarchitektur des PCs angesehen. Die Hauptplatine des NeXT-Computers kann man als Einplatinencomputer bezeichnen, auf dem die überwiegend in Aufsetztechnik gefertigte Elektronik dicht gepackt untergebracht ist. Laut Jobs „das kompakteste Board auf der Welt". Auf dieser Leiterkarte befinden sich zunächst die fünf Prozessoren: als CPU der 32-Bit-Prozessor *Motorola 68030*, der mit 25 MHz arbeitet, der mathematische Koprozessor *Motorola 68882* und ein 24-Bit-Festkomma-Signalprozessor *Motorola DSP 56001* zur Hochgeschwindigkeitsverarbeitung von speziellen mathematischen Berechnungen, Bildern, Signalen oder bei der Sprach- und Musiksynthese (lt. NeXT etwa 50- bis 100mal schneller als die Paarung 68030 + 68882). Dazu kommen der DMA-Chip *Integrated Channel Processor* (ICP) und der Disk-Chip *Optical Storage Processor* (OSP), die von NeXT als anwendungsspezifische Schaltkreise entwickelt wurden. Weiterhin ist auf der Leiterkarte unter anderem der 8-MByte-RAM enthalten, wobei noch Steckleisten für den Einsatz von weiteren acht 1-MByte-Modulen vorhanden sind. Bei Verwendung von 4-MByte-Modulen (SIMMs – Single Inline Memory Modules) wären somit künftig 64 MByte Hauptspeicher möglich. Schließlich sind auf der Platine die über die Gehäuserückseite zugänglichen Schnittstellen untergebracht: Monitorport für Video, Maus, Sound usw., serielles Druckerinterface, SCSI (Small Computer System Interface) zum Anschluß der Massenspeicher mit 4 MBit/s, 2 weitere serielle Schnittstellen, eine Ethernet-LAN-Schnittstelle sowie der DSP-Anschluß für ein Modem oder Fax-Gerät. Dank dieses Hardwareaufbaus und der Tatsache, daß Steven Jobs versucht hat, die Architektur einiger Großrechner nachzubilden, bei der separate Prozessoren dafür sorgen, daß die CPU nicht mit zeitaufwendigen „Nebenaufgaben" wie Ein-/Ausgabe behelligt wird, soll gegenüber bisherigen PCs die Leistung des Gesamtsystems mit der des Prozessors von 5 MIPS (Millionen Instruktionen pro Sekunde) übereinstimmen. Auch in Sachen Peripherie hat der NeXT-Computer Besonderes zu bieten. So ist er der erste Rechner, der als Serienausstattung ein magnetooptisches Laufwerk besitzt, das mit lösch- und wiederbeschreibbaren Platten in Cartridges als wechselbare Datenträger arbeitet. Die Kapazität liegt bei beachtlichen 256 MByte, die mittlere Zugriffszeit allderdings nur bei etwa 65 bis 100 ms. Als Nachteil könnte sich in Zukunft erweisen, daß sich die Firma Canon, Hersteller des Laufwerks, nicht an den voraussichtlichen Standard der ISO gehalten hat und es somit zu Kompatibilitätsproblemen gegenüber anderen Herstellern kommen kann. Zusätzlich zu diesem Laufwerk kann ein weiteres optisches Laufwerk oder – seit einiger Zeit – auch ein herkömmliches Festplattenlaufwerk (bis zu 670 MByte) verwendet werden. Als Manko ist bisher noch das Fehlen eines Diskettenlaufwerkes zu betrachten; schließlich liegen die Programme der meisten Anwender noch in dieser Form vor. Jobs ist in dieser Frage seiner Zeit vielleicht doch etwas zu weit vorausgeeilt. Zukunftsträchtig ist mit Sicherheit die von NeXT entwickelte objektorientierte Softwareumgebung. So wurde unter anderem eine extrem leicht zu erlernende grafische Benutzeroberfläche entwickelt, die selbst gegenüber der Apple HyperCard Vorteile aufweisen soll. Sie liegt quasi über dem ansonsten recht anwenderunfreundlichen Betriebssystemkern – Mach genannt –, der eine Variante von Unix ist. Die Firma IBM hat diese Bedienoberfläche für interessant genug gehalten, um sich für 10 Millionen Dollar die Lizenzrechte daran zu sichern.

Der NeXT-Computer bietet als weitere Erleichterung für den Nutzer übrigens erstmals eine einheitliche Kommandosprache für den Laserdrucker und die Bildschirmausgabe. Verwendet wird das sogenannte Display-PostScript, so daß auf dem Bildschirm wirklich das zu sehen ist, was der Drucker auf das Papier bringt. Allerdings ist die Auflösung des Monitors mit 94 Punkten/Zoll (1120 × 832 Punkte) geringer als die des Laserdruckers (400 Punkte/Zoll). Schließlich konnten dem Computer dank des Digitalen Signalprozessors herausragende Leistungen in bezug auf Musiksynthese sowie Spracheingabe und -ausgabe in Echtzeit mitgegeben werden. Die Systemsoftware unterstützt bereits Sprache und Musik; die vom Computer gelieferte Tonqualität entspricht CD-Standard.

Zusammengefaßt: Steven Jobs ist es gelungen, mit seiner Mannschaft auf der Basis modernster Hardware und des multitasking- und multiuserfähigen Betriebssystems Mach, einschließlich einer fortschrittlichen Benutzeroberfläche, einen Computer mit herausragenden Grafik- und Echtzeitverarbeitungsfähigkeiten zu entwickeln.

MP-We

NuBus Eurokarte Typ C Backplane-Verbindung
68030-CPU
Speicherpuffer
8 MByte RAM
VLSI-DMA-Chip
68882-Coprozessor
VLSI-Disk-Chip
Bipolar-Array für serielle Video- u. Ethernet-Daten
Video-Monitor-Port
„Thin"-Ethernet-Port
Serieller Laserdrucker-Port
5390-SCSI-Controller
Serielle Mini-DIN-8-Ports
DSP-Port
DSP56001
128 KByte PROM
32 KByte SRAM
Verbindung für optisches Laufwerk



Schnelles Löschen beliebiger Fenster

für den KC 87...

Bei den Kleincomputern KC 87 und Z9001 gab es bisher in Basic nur die einfache, relativ langsame CLS-Routine des Interpreters. Die in der MP 7/88 vorgestellte schnelle Lösung schloß immer das vollständige Löschen des Bildschirms ein. Im folgenden soll eine Variante vorgestellt werden, die es ermöglicht, beliebige Fenster schnell zu löschen.

Aus den Arbeitszellen des Betriebssystems für die Fenstergröße werden die entsprechenden Daten entnommen und daraus die Anzahl der Schleifendurchläufe berechnet. Zu Beginn wird der Kursor ausgeblendet und am Ende der Routine auf der entsprechenden linken oberen Position wieder gesetzt.

Das Programm arbeitet auf jedem Adreßbereich. Soll der Bildschirm nicht mit Leerzeichen beschrieben werden, muß in der der Speicherzelle <anfangsadresse> +68 der ASCII-Code des entsprechenden Zeichens stehen. Folgende Arbeitszellen werden geprüft:

```
3E 20                LD A,32            62,32
32 C8 EF             LD (OEFC8H),A      50,200,239
21 00 EC             LD HL,OEC00H       33,0,236
11 28 00             LD DE,28H          17,40,0
DD 21 3B 00          LD IX,3BH          221,33,59,0
DD 7E 00             LD A,(IX)          221,126,0
4F                   LD C,A             79
FE 00                CMP 0              254,0
28 04                JRZ M1-#           40,4
19            M0:    ADD HL,DE          25
3D                   DEC A              61
20 FC                JRNZ M0-#          32,252
DD 23         M1:    INC IX             221,35
51                   LD D,C             81
DD 7E 00             LD A,(IX)          221,126,0
3D                   DEC A              61
91                   SUB C              145
47                   LD B,A             71
DD 23                INC IX             221,35
DD 7E 00             LD A,(IX)          221,126
4F                   LD C,A             79
5F                   LD E,A             95
14                   INC D              20
1C                   INC E              28
D5                   PUSH DE            213
16 00                LD D,0             22,0
85                   ADD L              133
6F                   LD L,A             111
30 01                JRNC M2-#          48,1
24                   INC H              36
DD 23         M2:    INC IX             221,35
DD 7E 00             LD A,(IX)          221,126,0
3D                   DEC A              61
91                   SUB C              145
4F                   LD C,A             79
3E 28                LD A,28H           62,40
91                   SUB C              145
5F                   LD E,A             95
79                   LD A,C             121
4F            M3:    LD C,A             79
36 20         M4:    LD (HL),32         54,32
23                   INC HL             35
0D                   DEC C              13
20 FA                JRNZ M4-#          32,250
19                   ADD HL,DE          25
10 F6                DJNZ M3-#          16,246
D1                   POP DE             209
0E 12                LD C,12H           14,18
CD 05 00             CALL 5             205,5,0
3E 10                LD A,10H           62,16
32 C8 EF             LD (OEFC8H),A      50,200,239
C9                   RET                201
```

Zeile Fensteranfang: 3BH (59)
Zeile Fensterende : 3CH (60)
Spalte Fensteranfang: 3DH (61)
Spalte Fensterende : 3EH (62)

Hier ist zu beachten, daß die Endpositionen immer die Anzahl der Spalten bzw. Zeilen angeben und dazu noch 1 addiert wird. Eine weitere Bedingung ist: ZE ≥ ZA; SE ≥ SA.

Das bedeutet für das Normalfenster:
ZA=0 SA=0 ZE=25 SE=41

Für Basic-Fans ist statt des Kommentars zum Listing rechts der Dezimalcode des Programms angegeben.

Uwe Schöne

... und den KC 85/2 und /3

Die Eingabe des Programms erfolgt mit dem Systemkommando MODIFY, bei der jedoch die Prüfsumme nicht berücksichtigt wird! Folgende Arbeitsschritte sind abzuarbeiten:

① Basic starten – –> verlassen mit BYE
② Eingabe der Maschinenroutine
③ Sicherheitskopie anfertigen mit SAVE 0400 06CF
④ %MODIFY 03D7
03D7 CC
03D8 06.
⑤ %REBASIC
⑥ >LIST

Nach korrekter Eingabe müßte jetzt nach einigem *Wirrwarr* der Bildschirm gelöscht werden und folgende Zeile zu sehen sein:

O CALL*40F

Dieser Maschinenunterprogrammaufruf ist notwendig, um das Programm zu initialisieren. Danach kann mit CLS oder SHIFT-HOME das aktuelle Fenster schnell gelöscht werden. Will man sein BASIC-Programm mit dieser Schnellöschroutine ausrüsten, so lädt man es jetzt am besten nach und lagert dann beide Programme zusammen (mit CSAVE"...") aus.

Wenn man jetzt mit NEW ein neues Programm bearbeiten will, sollte man die Schnellöschroutine mit CALL*442 wieder ausschalten, um undefinierte Zustände des Computers zu vermeiden. Beide CALL-Befehle sind auch im CAOS-Menü enthalten (WCLSON bzw. WCLSOFF).

```
0400 00 BF 06 00 00 9C 7F 7F  025F
0408 57 43 4C 53 4F 4E 01 F5  02CC
0410 E5 C5 DB 88 47 CB D7 D3  05C9
0418 88 21 4D 04 22 B9 B7 21  02AD
0420 55 04 22 C4 B7 2A B0 B7  0387
0428 7E 32 60 04 23 7E 32 61  0248
0430 04 78 D3 88 C1 E1 F1 C9  0533
0438 7F 7F 57 43 4C 53 4F 46  02CC
0440 46 01 E5 F5 DB 88 6F CB  04BE
0448 D7 D3 88 7D CD 03 F0 20  048F
0450 D3 88 F1 E1 C9 FE 0C C2  05C2
0458 5F 04 3A A2 B7 CB 5F C2  03E2
0460 FF FF C5 D5 E5 F5 DD 7E  06CD
0468 0D F5 08 D9 C5 D5 E5 F5  0557
0470 D9 08 2A 9E B7 7C 85 B7  0418
0478 CA E6 05 21 8D 04 11 8E  0306
0480 04 01 22 01 05 36 D5 ED  0225
0488 B0 18 27 41 20 50 52 4F  0241
0490 44 55 43 54 49 4F 4E 20  0236
0498 42 59 20 20 2A 2A 2A 20  0179
04A0 42 2E 50 2E 53 2E 43 2E  01E0
04A8 20 20 2A 2A 2A 20 42 6C  018C
04B0 6E 2E 21 9E B7 3A 9C B7  039F
04B8 FE 20 30 0F 86 FE 21 30  0332
04C0 14 7E CD 63 06 CD 45 06  02E0
04C8 C3 EE 04 7E CD 7B 06 CD  044E
04D0 45 06 D9 18 19 D6 20 CD  0318
04D8 6F 06 3E 20 E5 2B 2B 96  02A4
04E0 CD 57 06 D9 E1 CD 45 06  03FC
04E8 D9 2E 1F CD 4B 06 D5 D9  03F2
04F0 D5 AF 57 5F 23 D9 57 5F  03EC
04F8 ED 53 A0 B7 23 3A 9F B7  044A
0500 47 3A 9D B7 C5 F5 CD F7  0553
0508 05 F1 3C CB 47 28 08 01  0275
0510 60 01 B7 ED 42 18 05 01  0265
0518 20 01 05 09 D9 F5 E6 07  02EA
0520 28 0D F1 CB 47 20 10 01  0269
0528 D8 01 B7 ED 42 18 0E F1  03D6
0530 01 08 01 05 09 18 06 01  0037
0538 60 01 B7 ED 42 D9 C1 10  03F1
0540 C3 D9 D1 D9 D1 3A A3 B7  05AB
0548 EB 57 5F D9 EB 57 5F 23  043E
0550 D9 E5 21 FD 05 36 02 21  033A
0558 03 06 36 20 21 08 06 36  00C4
0560 20 21 3A 06 22 15 06 E1  019F
0568 23 3A 9F B7 47 3A 9D B7  0388
```

```
0570 C5 F5 CD F7 05 F1 3C 09  04B9
0578 D9 F5 E6 07 28 11 F1 CB  04B0
0580 47 20 09 01 58 01 05 B7  0186
0588 ED 42 18 09 09 18 06 F1  0268
0590 01 08 01 05 09 D9 C1 10  01C2
0598 D7 21 08 06 36 80 21 FD  02DA
05A0 05 36 04 21 03 06 36 80  011F
05A8 21 1A 06 22 15 06 2A 9C  0144
05B0 B7 EB CD 03 F0 32 01 01  0396
05B8 B2 0D B7 ED 42 CB 2C CB  0467
05C0 1D 09 3A 9E B7 3D 4F 06  0247
05C8 01 05 3A 9F B7 54 5D 13  025A
05D0 C5 E5 F5 AF 77 79 B7 28  051D
05D8 02 ED B0 F1 E1 01 28 01  039B
05E0 05 09 C1 3D 20 E7 D9 08  02F4
05E8 F1 E1 D1 C1 D9 08 F1 DD  0613
05F0 77 0D F1 E1 D1 C1 C9 ED  059E
05F8 73 8B 04 F3 3E 04 08 3E  027D
0600 02 08 01 80 01 05 D9 01  016B
0608 80 01 05 B7 ED 42 D9 F9  043E
0610 C3 8D 04 3D CA 1A 06 09  0284
0618 18 F5 08 3D 28 1B 08 01  019E
0620 60 01 B7 ED 42 3E 04 01  028A
0628 80 01 05 D9 01 E0 01 B7  02F8
0630 ED 42 01 80 01 05 D9 18  02A7
0638 D6 08 ED 7B 8B 04 FB C9  0499
0640 09 F9 C3 A3 04 3A 9C B7  03F9
0648 86 3D 6F 3A 9D B7 07 07  02CE
0650 07 67 CD 03 F0 34 C9 E5  0410
0658 21 8D 04 11 40 06 CD 87  025D
0660 06 E1 C9 E5 21 A3 04 11  036E
0668 13 06 CD 94 06 18 E9 E5  0366
0670 21 A3 04 11 13 06 CD 87  0246
0678 06 E1 C9 E5 21 8D 04 11  0358
0680 40 06 CD 94 06 18 E9 FE  03AC
0688 01 28 13 B7 1F 38 21 4F  01BA
0690 06 01 05 09 36 D9 23 36  017D
0698 C3 23 73 23 72 C9 36 2B  0318
06A0 23 36 2B 23 36 5E 23 36  0194
06A8 23 23 36 23 23 23 18 E4  01E1
06B0 4F 06 01 05 09 36 33 23  00F0
06B8 36 D5 23 18 D7 0C 00 CA  02F3
06C0 06 00 00 9D 20 AE 34 30  01D5
06C8 46 00 00 00 00 00 00 00  0046
06D0 00 00 00 00 00 00 00 00  0000
```


Literatur

/1/ Schlenzig, S.; Schlenzig, K.: Tips und Tricks für kleine Computer. Militärverlag, Berlin 1988, S. 103, 162

/2/ Langenhan, H.: Schnelles Bildschirmlöschprogramm für KC 85/3. Mikroprozessortechnik, Berlin 2 (1988) 5, S. 157

Volker Bartheld

Redabas-Tip

Zu einigen Detailproblemen

Die durch längere Arbeit mit Redabas gewonnenen Erkenntnisse zu einigen Detailfragen könnten von allgemeinem Interesse sein und besonders Nutzern, die noch nicht an die Grenzen dieses Sprachkonzeptes vorgestoßen sind sowie Anfängern unnötige Zeitverschwendung durch aufwendiges Austesten von zunächst unklarem Systemverhalten ersparen. Die Darlegungen erheben dabei allerdings keinerlei Anspruch auf Vollständigkeit. Die angesprochenen Probleme resultieren zum großen Teil aus fehlenden oder schnell zu überlesenden Informationen in der Redabas-Beschreibung/∧/, so daß sich Programmfehler unnötigerweise einschleichen. Hinzu kommt, daß in einigen Fällen keine Fehlerausschriften erscheinen, was die Ursachenforschung erschwert.

Begrenzter Arbeitsspeicher

Die Tatsache, daß unter Redabas nur 64 temporäre Speichervariablen zulässig sind, erweist sich bei größeren Programmen schnell als ungünstig. Kommt es darauf an, zunächst möglichst viele schnell abrufbare Informationen zur Verfügung zu haben, um sie erst später zu verarbeiten oder auszugeben, so bietet sich die in /2/ vorgeschlagene Methode an. Alle Informationen werden auf diese Weise im RAM verwaltet, was Programmlaufzeit einspart (keine Diskettenzugriffe).

Unter Beachtung der Leistungsparameter von Redabas (/1/) könnte man annehmen, pro Variable maximal 254 Zeichen entsprechend der angegebenen max. Zeichenkettenlänge zu speichern, also theoretisch 64 × 254 Zeichen. In /3/ wird dies unter Punkt 5.1. ausdrücklich bestätigt "... maximale Länge jeweils 254 Byte, maximale Zahl 64".

Ich wollte dieses Verfahren für 10 Variablen mit je 250 Zeichen nutzen. Während der Programmerprobung zeigte sich jedoch, daß oberhalb einer bestimmten Zeichenzahl das Programm ohne Fehlerausschrift verlassen wurde. Daraufhin entstand ein Testprogramm (Bild 1), das analog /4/ über Indizes die Variablen RGL1, RGL2, RGL3 ... usw. anlegt und auf diesen je nach Abbruchkriterium IF ZCH<251 (bzw. 101) 250 (bzw. 100) Zeichen speichert.

```
* TEST VARIABLE/BYTE
SET TALK OFF
ERASE
STORE 1 TO ZZ
STORE '1' TO ZHLR
STORE T TO SCHL
STORE 'x' TO RGL1
STORE 1 TO ZCH
DO WHILE SCHL
IF ZCH < 251
STORE RGL&ZHLR + 'x' TO RGL&ZHLR
STORE ZCH + 1 TO ZCH
@ 9,0 SAY ZHLR + ' ' + STR(ZCH,3,0)
ELSE
STORE 1 TO ZCH
STORE ZZ + 1 TO ZZ
IF ZZ < 10
STORE STR(ZZ,1,0) TO ZHLR
STORE 'x' TO RGL&ZHLR
ELSE
STORE STR(ZZ,2,0) TO ZHLR
STORE 'x' TO RGL&ZHLR
ENDIF
ENDIF
ENDDO
RETURN
```

***Bild 1** Testprogramm zur Variablenbelegung und zur Variablenanzahl*

Der Programmabbruch erfolgte ohne Fehlerausschrift bei:

- 250 Zeichen/Variable nach 4 vollständig und einer mit 167 Zeichen beschriebenen Variablen. Das ergab einschließlich der Steuervariablen: 9 temporäre Variablen, die insgesamt 1190 Bytes lang waren.
- 100 Zeichen/Varibale nach 13 vollständig und einer mit 81 Zeichen beschriebenen Variablen. Es ergab einschließlich der Steuervariablen: 18 temp. Variable 1414 Bytes lang.

Offensichtlich ist der Redabas-Arbeitspeicher begrenzt, wobei der verfügbare Speicherplatz von der Menge der auf den temporären Variablen abgelegten Informationen abhängig ist. Leider findet sich dazu in /1/ kein Hinweis. Damit ist die in /2/ angegebene – unter Redabas-Bedingungen effektive – Methode in größerem Umfang nicht anwendbar.

Variablendateien (.VAR)

Unter Beachtung der dargelegten Erfahrungen ist die in /1/, Punkt 5.3.3. getroffene Aussage verwirrend: „Die Variablendatei kann maximal 64 Speichervariable – jede bis zu 254 Zeichen lang – aufnehmen". Diese Aussage wäre nur dann sinnvoll, könnten die Variablen nach und nach ausgelagert werden. Aus der Befehlsbeschreibung sind dazu keine exakten Anhaltspunkte zu entnehmen. Ein Test ergab das in Bild 2 dargestellte Ergebnis.
Damit ist klar: Die zuvor auf der Datei SPEICHER.VAR befindlichen Variablen werden bei jeder weiteren Auslagerung durch die aktuellen Variablen überschrieben. Die *Additive*-Option des RESTORE-Befehls ist zum Verhindern des Überschreibens hier leider nicht anwendbar. Es bleibt dadurch völlig im Dunkeln, wie die angegebene maximale Speicherkapazität von 64 × 254 Zeichen ausgenutzt werden kann.

Die Funktion VAL

Laut /1/ werden bei der Umwandlung eines Strings in einen numerischen Wert der Dezimalpunkt und nachfolgende Stellen abgeschnitten. Ohne Benutzung der Makroersetzung können auf die in Bild 3 angegebene Weise die nachfolgenden Stellen gerettet werden.

Löschen temporärer Variablen

Mitunter ist es von Vorteil, nur die Variablen anzugeben, die nicht gelöscht werden sollen. Dann ist der Befehl, in der Form

RELEASE ALL EXCEPT <muster>

anzuwenden. Was aber, wenn sich die Variablen nicht in ein Muster einordnen lassen?
Die syntaktisch falsche Notation A1, B2, C3 (kein Muster) wird ohne Beanstandung in allerdings unerwarteter Weise (Bild 4) abgearbeitet: alle Variablen gelöscht! Bei Angabe einer Variablen wird der Befehl exakt abgearbeitet. In derartigen Fällen ist damit mehrzeilig vom RELEASE-Befehl Gebrauch zu machen.

```
STORE '1234.56' TO SWERT
STORE VAL($(SWERT,1,4) + $(SWERT,6,2)) * 0.01 TO NWERT
? NWERT
1234.56
```

Bild 3 Wandlung einer Zeichenkette in einen numerischen Wert

Gebrauch der Substring-Funktion

STORE 'x' TO ZCH
? $ (ZCH,2,4)
***** UNZULAESSIGE ZEICHENKETTE!**

Die Fehlerursache der hier dargestellten Situation ist bei größeren Programmen durchaus nicht sofort erkennbar. Bei der routinemäßigen Benutzung der Substring-Funktion, beispielsweise innerhalb einer DO WHILE-Schleife sollte streng darauf geachtet werden, daß die aktuelle Länge der Zeichenkette mindestens der Summe <Start> + <Länge> der Substring-Parameter entspricht. Interessanterweise ergab der obige Befehl schon nach der folgenden Änderung keine Fehlerausschrift mehr:

STORE 'x' TO ZCH
? $ (ZCH,1,4)
x

Löschen der aktivierten Eingabemasken

In der mehrseitigen Beschreibung zum @-Befehl findet sich in /1/, S. 133 der wichtige Hinweis, daß die Anzahl der gleichzeitig aktivierbaren GET-Masken auf 64 beschränkt ist. Wird dies vom Programmierer übersehen, kommt es beim Überschreiten dieser Grenze zu **keiner** Fehlerausschrift. Vielmehr werden einige Kursor-Steuerungen (z. B. Cursor rückwärts) wirkungslos. In so einer Situation kann sich besonders beim Anfänger das Suchen der Ursachen als sehr zeitaufwendig gestalten. Besonders bei sich zyklisch wiederholenden Eingaben innerhalb einer Schleife sind schnell die 64 aktivierbaren GET-Masken erreicht. Auch bei weniger umfangreichen Eingaben sollten nicht mehr benötigte GET-Masken rechzeitig mit CLEAR GETS entaktiviert werden. Das ist besonders dann nötig, wenn ein neues Eingabemenue angeboten wird. Ohne Entaktivierung ist es dann möglich, von der ersten Eingabeposition aus über die Kursortasten ↑ bzw. ← die vorherige Eingabeposition des *alten* Menuebildes zu erreichen; der Kursor steht dann scheinbar undefiniert irgendwo in dem aktuellen Menuebild.

```
STORE 1 TO W1
STORE 2 TO W2
STORE 3 TO W3
SAVE TO SPEICHER
CLEAR
DISPLAY MEMORY
** SUMME ** 00 TEMP. VARIABLE 00000 BYTES LANG
STORE 10 TO A1
STORE 11 TO A2
STORE 12 TO A3
SAVE TO SPEICHER
CLEAR
RESTORE FROM SPEICHER
DISPLAY MEMORY
A1      (N)    10
A2      (N)    11
A3      (N)    12
** SUMME ** 03 TEMP. VARIABLE 00021 BYTES LANG
```

Bild 2 Test mit Variablendateien

```
STORE 1 TO A1
STORE 2 TO B2
STORE 3 TO C3
STORE 4 TO D4
RELEASE ALL EXCEPT A1,B2,C3
DISPLAY MEMORY
** SUMME ** 00 TEMP. VARIABLE 00000 BYTES LANG
```

Bild 4 Wirkung des Befehls RELEASE ALL EXCEPT

Literatur

/1/ VEB Robotron-Projekt Dresden: Programmtechnische Beschreibung REDABAS. Dresden 1985
/2/ Svenson, G.: Nutzung einer Zeichenkettenvariablen zur gleichzeitigen Abspeicherung verschiedener Werte. Mikroprozessortechnik 1 (1987) 5 S. 149
/3/ Weller, T., Donner, M.: Kurs REDABAS, Arbeit mit Datenbanken Mikroprozessortechnik 1 (1987) 8 S. 239
/4/ Matzke, B.: Indizierte Variablen unter REDABAS. Mikroprozessortechnik 4/1988 2 (1988) 4 S. 104

Joachim Heinze

A3-Druckertreiber für den A 7100

Besteht der Wunsch, die A4-Druckertreiber K631 3HR8.SYS und K631LR8.SYS auch für A3- oder A2-Druck zu verwenden, sind dazu lediglich die Eintragungen für die Anzahl der Pixel in x- und y-Richtung zu ändern. Durch die Suche der entsprechenden Originalwerte (jeweils 2 Byte), kann man diese Eintragungen finden. Ihre Änderung ist mit dem Turbo-Pascal-Programm **Patch** möglich.
Wir haben für die Erstellung eines A3-Treibers folgende Pixel-Zahlen getestet: Pixelanzahl in X-Richtung: 1499 und in Y-Richtung (Originalwert): 1367. Durch Veränderung dieser Werte können beliebige Formate bis hin zum A2-Format eingestellt werden (X: 1499; Y: 2240).
Man sollte nicht vergessen, den unter einem anderen Namen neu erstellten Treiber in die Datei ASSIGN.SYS einzutragen.

G. Tschuch, H. Nieber

```
program patch;

const
   blocknummer = 112;
   bytenummer  = 42;
   { Treiberversion vom 1.12.1985 }

var
   f             : file;
   name          : string[30];
   i,j,k         : integer;
   puffer,dummy  : array[0..127] of byte;

begin
      repeat
       write('Name des zu patchenden Files (K6313HR8.SYS z.B.): ');
       readln(name);
       assign(f,name);
       {$I-}reset(f){$I+};
      until ioresult=0;
      for i:=0 to blocknummer do blockread(f,puffer,1);
      reset(f);
      for i:=0 to blocknummer-1 do blockread(f,dummy,1);
      write('Anzahl der Pixel in X-Richtung: ');
      readln(i);
      puffer[bytenummer]:=lo(i);
      puffer[bytenummer+1]:=hi(i);
      write('Anzahl der Pixel in Y-Richtung: ');
      readln(i);
      puffer[bytenummer+2]:=lo(i);
      puffer[bytenummer+3]:=hi(i);
      blockwrite(f,puffer,1);
      close(f);
end.
```

Kleines Lexikon der Mikrorechentechnik

O wie Operationsteil



Zeichnung: Dahmen



Börse

Wir installieren Ihre Festplatte

Eine Lösung zur Organisation und Verwaltung der Festplatte unter MS-DOS bietet das Harddisk-Management-System, das auf der in MP 12/88 im Beitrag „Festplattenorganisation – Ein Bibliothekskonzept für die Harddisk unter DOS" vorgestellten Idee beruht.
Das System besteht aus einem Satz von Batch-Dateien, die durch verschiedene Utilities zur Speicherverwaltung und Batch-File-Programmierung ergänzt werden. Die offene Struktur bietet eine hohe Flexibilität, die das *Aufsetzen* weiterer Benutzeroberflächen (beispielsweise NC, PMM) problemlos ermöglicht und jedem Nutzer gestattet, das vorgegebene System nach seinen individuellen Bedürfnissen zu erweitern. Im einzelnen bringt die Festplattenverwaltung folgende Vorteile:
– multivalente Nutzbarkeit (projekt- oder nutzerorientiert)
– automatisiertes Backup von Nutzerbereichen (SYS1 .. SYSn)
– optimale Anpassungsmöglichkeiten an das jeweilige Nutzer- und Nutzungsprofil durch flexible Systemarchitektur
– Unterstützung der Partitionierung der Harddisk in mehrere logische Laufwerke mit identischer Architektur
– Erweiterungsmöglichkeiten für Paßwort- und Zugriffsschutz
– strikte Trennung von Programmen und Daten über Nutzer- und Arbeitsverzeichnisse
– optimierter Zugriff über Batch-Files und RAM-Disk
– Management für speicherresidente Programme
Das Harddisk-Management-System ist für den AC 7150, EC 1834 und kompatible Geräte sowie für AKTs verwendbar.

Technische Universität Dresden, Sektion 20/WE Mommsenstr. 13, Dresden, 8027; Tel. 2326118

Hanisch

Turbo-Editor unter WEGA

Zur Effektivierung der Programmentwicklung unter WEGA wurde von uns ein Editor mit den Leistungsmerkmalen der Turbo-Pascal-Editoren entwickelt und auf P 8000-Computern implementiert. Der Editor ist eine C-Implementation der Public-Domain-Software *Turbo-Pascal-Editor-Toolbox* und zeichnet sich durch folgende Leistungsmerkmale aus:
– WordStar-Kompatibilität in der Kommandosyntax; der Kommandovorrat entspricht dem Non-Document-Mode von WordStar.
– hohe Editiergeschwindigkeit durch ausschließliche Bearbeitung über dynamische Speicherbelegung (keine Zwischendateien auf Diskette); damit ist die maximale Größe einer zu bearbeitenden Datei zwar durch die Größe des nutzerseitig zur Verfügung stehenden Hauptspeichers begrenzt, praktisch konnten jedoch schon 200-KByte-Textdateien problemlos editiert werden.
– beschleunigte Bildschirmaus- und Tastatureingabe durch Anwendung der ESC-Sequenzen für VT100- und ADM31-kompatible WEGA-Terminals; unter DCP werden die schnelleren BIOS-Interrupts direkt angesprochen.
– nützliche Funktionserweiterungen, wie AutoIndent-Funktion und File-Pick-Up entsprechend der ALT-F3-Funktion der Turbo-Pascal-5.0- und TURBO-C-2.0-Editoren, verfügbar.
– Einbindung des Editors (auch in durch bedingte Compilierung minimierten Versionen) in größere Softwareprojekte problemlos möglich.
– volle Lauffähigkeit unter WEGA und DCP sowie gute Portabilität des durchgängig in der Sprache C geschriebenen Editors gesichert.
Wir übermitteln Interessenten innerhalb von 14 Tagen nach Zusendung einer WEGA-tar- sowie einer 720-KByte-DCP-Diskette:
– eine kostenlose *Mini-Demoversion* des Editors mit integriertem TUTOR und kompletter Dokumentation, geeignet zur Bearbeitung von Dateien von bis zu 20 KByte unter WEGA und DCP
– in Kopplung mit der Mini-Demoversion ein Vertragsangebot zur Nachnutzung des C-Quellkodes des Editors.

Ingenieurhochschule für Seefahrt Warnemünde/Wustrow, Sektion Schiffsführung/WB FE, Warnemünde, 2530

Dr. Klabunde

Dateikonvertierung am P8000

Zur Gewährleistung des Datenaustausches zwischen den zur Zeit vorherrschenden Betriebssystemen entstand ein Dateikonvertierungsprogramm. Das Programm **conv** arbeitet bildschirmorientiert auf dem P8000 und gestattet den Datenaustausch zwischen den Formaten der Betriebssysteme MS-DOS, CP/M und UNIX.
Auf zwei Fenstern werden die Inhaltsverzeichnisse zweier unterschiedlicher Dateiformate dargestellt. Mit den Kursortasten und der Returntaste lassen sich Dateien markieren und anschließend in das jeweils andere Format konvertieren. Neben der Konvertierung besteht die Möglichkeit der Ausgabe der Dateien auf dem Bildschirm und des Löschens von Dateien. Für die Arbeit in Verzeichnishierarchien unterstützt das Programm ein Kommando zum Verzeichniswechsel.
Änderungen am WEGA-Betriebssystem des P8000 sind nicht erforderlich, das Konvertierungsprogramm benutzt nur die vorhandenen Diskettentreiber.

Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 5613240

Plass

Informationssystem für den KC 85/3

Für den KC 85/3 wurde das Programm **IMASTER** entwickelt. IMASTER dient zur dialoggeführten oder automatischen Darstellung einer geordneten Folge von pseudografischen Bildtafeln. Die Bildtafelfolgen können Tagungsprogramme, Übersichten zur Studienbewerbung oder ähnliches darstellen. Bildtafeln werden mit dem Bildeditor EDIGRAF (entwickelt an der Verkehrshochschule Dresden) erzeugt und in komprimierter Form auf einer Magnetbandkassette gespeichert. Die derzeitige Version des Programms IMASTER ist so angelegt, daß sich maximal 32 Bildtafeln gleichzeitig im Speicher des KCs befinden können. Zum Betrieb des Programms werden nur ein KC 85/3 und ein 64-KByte-Speichermodul benötigt.

Technische Universität „Otto von Guericke" Magdeburg, Sektion Informatik, WB-MI, PF 124, Magdeburg, 3010; Tel. 592766

Hinz

Funktionstastenkodierung und schnelle Alphanumerik am A 7100

Über die KGS-Firmware konnte eine erhebliche **Beschleunigung aller Löschvorgänge und Textanzeigen** erreicht werden. Das belegen die folgenden Zahlen:
① Bildschirm löschen in 80 ms. (Original: 2,8 s oder 0,67 s beim Zeilenhochrollen)
② Zeile löschen in 21 ms (Original: 112 ms)
③ Bildschirmfüllung = 2000 Zeichen in 1,0 s (Sparanzeige in 0,74 s); (Original: mit Blockkursor 4,1 s, mit Unterstrich-Kursor 2,8 s, Kursor unsichtbar 1,2 s)
Auch bei den Textanzeigen ist die Beschleunigung erheblich, da in der Praxis nicht mit unsichtbarem Kursor gearbeitet werden kann. Jetzt wurde der zeitaufwendige Ausgabekursor durch einen Eingabekursor ersetzt, wozu das Betriebssystem geringfügig anzupassen ist. Besonders drastisch wirkt sich die Beschleunigung bei typischen Betriebsweisen in dBase II aus. Für die feste KGS-Firmware wurde auch eine Version mit 36 Bildschirmzeilen erarbeitet.
Das Maschinenprogramm **PFTA.CMD** erlaubt unter CP/K in einfacher Weise, die PF-Tasten mit wählbaren Zeichenfolgen zu belegen und maximal 6 derartige Sätze von Tastenbelegungen zu speichern. Bei häufig auftretenden Zeichenfolgen (etwa die reservierten Worte einer Programmiersprache) lassen sich die Eingaben erheblich reduzieren. Bei komplizierten Zeichenfolgen sinken zusätzlich die Eingabefehler. Durch die rasche Umschaltmöglichkeit zwischen verschiedenen Tastensätzen ist die Beschränkung auf nur 12 einstellbare Tasten gemildert. Gegenüber CPKKEY.CMD ergeben sich mehrere Vorteile:
① Zusätzlich zum Systemspeicher, dessen Inhalt beim Booten aktiviert wird, sind 5 weitere Speicher für komplette PF-Tasten-Belegungssätze vorhanden. Das Abspeichern der aktuellen Belegung und das Reaktivieren aller sechs gespeicherten Sätze ist jederzeit und ohne Neubooten möglich.
② Die Stringlänge pro PF-Taste ist nicht auf 9 Zeichen begrenzt. Allerdings geht die Erweiterung auf Kosten der Folgetasten. Im Extremfall kann die Taste PF1 einen String der Länge 119 Zeichen ausgeben.
③ Auch Steuerfolgen mit dem Zeichen ESC sind möglich.
④ Jede Änderung wird sofort ohne Neubooten wirksam (ähnlich BREAK-PFx)
⑤ Die aktuelle Tastenbelegung kann jederzeit angezeigt werden.
⑥ Zur temporären Einstellung kann auf die Methode BREAK-PFx prinzipiell verzichtet werden. Das ist wichtig, wenn etwa wegen laufender Zeitinterrupts die BREAK-Taste nicht benutzt werden darf.
⑦ Kurzes 2-KByte-Maschinenprogramm.

Akademie der Wissenschaften der DDR; Zentralinstitut für Astrophysik, Sternwarte Sonneberg, Postfach 55–27, Sonneberg, 6400; Tel 2287

Dr. Fürtig

EPROM-Programmiergerät mit Schnittstelle V.24

Das EPROM-Programmiergerät **EPR189** ist ein selbständiges Gerät mit eigener Stromversorgung und enthält als Kernstück einen EMR UB 8840. Anschließbar ist es über eine Standardschnittstelle V.24 an jeden Rechner. Die Bedienung erfolgt über eine Funktionstastatur. Programmierbar sind die EPROM-Typen 2716 bis 27512. Das EPR189 besitzt folgende Betriebsarten:
1. Betriebsarten, die einen Host-Rechner erfordern: Programmieren und Auslesen eines EPROMs. Der Datenaustausch erfolgt dabei über Intelblöcke.
2. Betriebsarten, die ohne Host-Rechner möglich sind:
– BLANC-Test eines EPROMs
– Duplizieren eines EPROMs
– Vergleichen von zwei EPROMs
Zur Realisierung der letzten beiden Betriebsarten sind auf dem Programmiergerät 2 Schwenkhebelfassungen angeordnet. Programmiert werden kann mit 50-ms-Impulsen oder mit dem intelligenten Programmieralgorithmus mit 1-ms-Impulsen. Fehler bei der Programmierung werden mit LEDs angezeigt. Mit dem EPR189 ist es also mit jedem Rechner über die Standardschnittstelle V.24 möglich, EPROMs zu programmieren und auszulesen. Weiterhin ist es möglich, in der Klein- und Kleinstserienfertigung EPROMs ohne Host-Rechner zu programmieren, indem von einem Mutter-EPROM Duplikate hergestellt werden können. 1990 wird eine 1. Kleinserie des EPR189 produziert. Gleichzeitig wird entsprechende Software für den Host-Rechner bereitgestellt, und ab Mitte 1990 stehen nachnutzungsfähige Unterlagen bereit.

Akademie der Wissenschaften der DDR, Zentrum für wissenschaftlichen Gerätebau, Schnellerstraße 138, Berlin, 1190; Tel. 6352311

Meinhardt

Erzeugnisübersichten

Die rechnergestützten *Erzeugnisübersichten* sind Bestandteil der Wissenschaftlich-technischen Konzep-

tion (WTK) des Kombinates Elektromaschinenbau (KEM). Die betrieblichen Daten (Erzeugnisdateien) werden auf Diskette an den Stammbetrieb, Bereich Technik, gegeben und dort zur *Erzeugnisübersicht des KEM* verdichtet und in vielfältiger Form ausgewertet. Die rechnergestützten Aussagen der WTK ermöglichen eine zielgerichtetere Einflußnahme des Generaldirektors und seiner verantwortlichen Leiter auf die Herausarbeitung der Strategie und der Hauptentwicklungsrichtungen in Wissenschaft und Technik, die Gewährleistung des notwendigen Erzeugnisumschlages gemäß vorgegebener normativer Produktionsdauer sowie die Absicherung der geforderten Effekte aus Forschung und Entwicklung, insbesondere Arbeitszeit-, Kosten- und Materialeinsparungen.
Darüber hinaus ist für den Entscheidungsprozeß in Wissenschaft und Technik verschiedener Leitungsebenen die Bereitstellung stets zugriffsbereiter aktueller Informationen zur Entwicklung des Erzeugnissortiments, zu den technologischen Aufgaben und zur Investstrategie gegeben. Gleichzeitig wurden Voraussetzungen für die Durchführung von Variantenrechnungen am ESER zur Optimierung des Erzeugnissortimets des KEM geschaffen.
Nutzbare Hardwarekonfigurationen sind die Computer: PC 1715, BC 5120/30, A 7100, A 7150 oder kompatible NSW-Technik sowie Drucker: K 6314, Epson FX 1000 oder andere Druckertypen mit Druckbreite ≧ 160 Zeichen/Zeile.
Die Programme des Programmpakets *WTK* sind in Basic geschrieben. Die Anpassung an kombinats- oder betriebsspezifische Bedingungen ist möglich und kann vereinbart werden.

VEB Kombinat Elektromaschinenbau, Abt. TWE 2, Hennigsdorfer Straße 25, Dresden, 8017; Tel 2245322

Kube

Mathematische Grundfunktionen für REDACOM-Programme

Das Programmsystem REDACOM beinhaltet standardmäßig nur eine geringe Anzahl von mathematischen Funktionen, diese reichen für die Lösung von Sachverhalten i. allg. nicht aus. Deshalb wurde eine Vielzahl weiterer mathematischer Funktionen aufbereitet, programmiert und in das Programmsystem integriert. Beim System REDACOM geschieht dies über UDFs (user defined functions) und beim System REDABAS-3 über Prozeduren. Neben den Standardfunktionen können Sie mit der Funktionsbibliothek folgende Funktionen lösen: Fakultät einer Zahl, Umrechnung von Gradmaß in Bogenmaß und zurück, alle trigonometrischen, zyklometrischen, Hyperbel- und Areafunktionen sowie Lösung des Gaußschen Fehlerintegrals, PI(), LN(), LG(), Signumfunktion und FRAC-Funktion.
Der Programmierung der aufgeführten Funktionen liegen z. T. Reihenentwicklungen zugrunde. Durch die Nutzung dieser Funktionen und der Makrooperation (mit dem Operationszeichen &) ist man in der Lage, einen Formelinterpreter für REDACOM bzw. für REDABAS-3 zu schaffen. Dieser Formelinterpreter kann vielfältig genutzt werden. So kann er beispielsweise in andere Programme eingefügt werden, um in Datenbanken abgelegte Formeln lösen zu können. Eine IF-THEN-ELSE-Verknüpfung kann ebenfalls über die Makrofunktion und den Formelinterpreter ausgewertet werden. Beispiele für die Formelnotationen:
- SIN(GB(45))
- SQRT(4.3*EXP(2.2))
- IF (3>X,SIN(PI()),45)*LN(2)

Technische Universität „Otto von Guericke" Magdeburg, Sektion Maschinenbau, WB Maschinenbautechnik, PSF 124, Magdeburg, 3010; Tel. 592670 oder 592672

Kretschmann/Pohlmann

KC 87 am ROLANET 1

Die Aufgabe bestand darin, die in der Abteilung Materialwirtschaft durch das Lagerbuchhaltungsprojekt anfallenden Informationen für einen größeren Nutzerkreis verfügbar zu machen. Dabei war der Echtzeitcharakter solcher Zahlen wie Kostenstellen- und Kostenträgerbelastungen, Materialbestände usw. zu berücksichtigen, was nur durch die maschinelle Übermittlung von Daten effektiv lösbar ist. Die Wahl fiel auf das lokale Netz ROLANET 1 als Übertragungsmedium. Als kostengünstiges und zugleich robustes Terminal wurde der KC 87 durch die Entwicklung eines abgerüsteten ROLANET 1-Controllers mit den Funktionen
- Taktgenerator
- Manchesterkodierung/-dekodierung
- Sender- und Empfängerstufen
- Taktrückgewinnung
- Kollisionsdekodierung
- Seriell-Parallel-Wandlung
- KC 87-Systembusanpassung

netzfähig gemacht.
Die Firmware realisiert die sichere Übertragung von Datenblöcken zwischen den LAN-Komponenten und ist im KC 87 in das BOS eingebunden. Der Nutzerzugriff erfolgt über OPEN-, READ-, WRITE- und CLOSE-Funktionen.

PGH Moderne Technik, F.-C.-Weiskopf-Straße 68, Dresden, 8027; Tel. 475071

Harazim/Reif

Grafischer Druck von Meßreihen und Funktionen

Vorrangig für Rechner ohne Bildschirmgrafik wurde ein Programm zur grafischen 2-D-Darstellung von Meßreihen und Funktionen entwickelt. Das Programm wurde in C geschrieben und ist unter Verwendung entsprechender Compiler weitgehend protabel. Es liegen beispielsweise Implementierungen für MC 80.22, CP/M-, CP/M-86- und MS-DOS-kompatible Rechner vor. Die Datenübergabe vom Anwenderprogramm an das Grafikprogramm erfolgt entweder über einen gemeinsamen Speicherbereich oder über eine Datei. Um eine Nutzung unabhängig von spezifischen Datenformaten unterschiedlicher Compiler zu ermöglichen, werden ausschließlich Integer- und Stringvariablen übergeben. Das Programm zeichnet sich durch folgende Eigenschaften aus:
- Darstellung mehrdeutiger Funktionen
- Darstellungsrichtung wählbar (horizontal/vertikal)
- maximale Auflösung bei Verwendung eines Druckers K 6311: 600 × 840 Pixel (auf A4-Format)
- Kurvendarstellung mit geschlossenem Linienzug
- Verbindung benachbarter Funktionswerte mit Hilfe einer Spline-Interpolation
- Darstellung einer beliebigen Kurvenzahl in einem Diagramm
- Markierung bestimmter Funktionswerte mit Symbolen
- Skalierung in beiden Koordinatenrichtungen automatisch nach Extremwerten oder nach vorgegebenen Grenzwerten.

Es können eine generierte Variante für die genannten Rechnertypen, der Quelltext sowie ein Beispielanwenderprogramm als Turbo-Pascal-Quelltext nachgenutzt werden.

Technische Hochschule „Carl Schorlemmer" Leuna-Merseburg, Hochschul-Industrie-Forschungsgruppe, Otto-Nuschke-Straße, Merseburg, 4200; Tel. 462597

Dr. Herrmann

Programme für Handwerks- und Handelsbetriebe

Für den Computer C 64 mit einem oder zwei Diskettenlaufwerken 1541 und Drucker Robotron K 6311 oder K 6313 haben wir folgende Programme für Handwerks- und Handelsbetriebe entwickelt, die wir zur Nachnutzung anbieten:
- Lohnrechnung für 50 Personen auf der Basis des Mehrlohnsystems, integrierte Stammdatenpflege, Ausdruck von Lohnzetteln, Lohnlisten, Monats- und Jahresabrechnung
- Journalführung für Bank/Postscheck/Kasse, Rechnungseingang, Rechnungsausgang, Geldein- und Geldausgang
- Lagerhaltung mit Zu- und Abgängen
- Rechnungslegung
- Energieabrechnung.

BESSER-JUNGPFLANZEN PSF 98, Halberstadt, 3600; Tel. 23121

Besser

Kopplung A 7150 – DZT 912 RS

Zur Kopplung des A 7150 unter DCP 1700 mit dem Digitalzeichentisch DZT 912 RS vom VEB Carl Zeiss JENA ist ein Treiber entwickelt worden. Die Kopplung erfolgt über die IFSS-Schnittstelle des Rechners (ASP-Karte). Eine andere Treibervariante bedient zum gleichen Zweck die USART-IFSS der ZVE-Karte. Programmiersprache ist Turbo-Pascal mit eingebundenem Assemblercode. Ausgerichtet ist das Treiberprogramm auf das Plotten von PCCAD-Plot-Dateien auf dem DZT. Es lassen sich auch alle anderen Dateien verarbeiten, die den Kommandovorrat des DZT beinhalten. Der Treiber gestattet u. a. eine relative Koordinatenstruktur und eine Koordinatendrehung der Zeichnungsformulare, die Unterbrechung der Zeichnungsarbeit mit der Möglichkeit, an gleicher Stelle zeitlich versetzt weiterzuarbeiten, und er läßt das mehrfache Übereinanderzeichnen (Ebenen, Ebenengruppen usw.) zu. Der Treiber ist dokumentiert und liegt zur Nachnutzung bereit.

VEB Ingenieurbüro Elektrogeräte, Abt. RRK, Markt 5, Karl-Marx-Stadt, 9010; Tel. 4415 6/332

Wagner

CP/M-Hardwareerweiterung mit zugehörigem BIOS

Durch den Einbau einer kleinen Leiterplatte mit 3 LS-TTL-Schaltkreisen und den Austausch von 8 Speicherschaltkreisen durch U 2164 kann für den ZX Spectrum und diverse Nachbauvarianten ein CP/M-Betrieb realisiert werden. Eine BIOS-Variante, die diese Hardware steuert, ermöglicht den Betrieb von CP/M.2.2 mit einem TPA von über 57 KByte. Hardwarevoraussetzung ist ein Disk-Interface mit TR-DOS 4.12 und mindestens ein Laufwerk. Das BIOS verwaltet ein Bildschirmformat von 64 × 24 Zeichen mit Hell- und Inversdarstellung, 2 Zeichensätze, eine Hardcopyfunktion, einen Tastaturpuffer, 2 Drucker und bis zu vier Laufwerke. Ein Datenaustausch mit SCP-Rechnern ist problemlos möglich. Die Software gestattet auch den Betrieb von Spectrum-Programmen unter CP/M.
Zum Nachnutzungsumfang gehören die unbestückte Zusatzplatine, Schaltplan, Einbauhinweise, das dazugehörige BIOS mit Beschreibung und Installationshinweisen sowie Software zum Formatieren, Installieren und für den Datenaustausch.

Akademie der Wissenschaften der DDR, Zentralinstitut für Elektronenphysik, Patentbüro, Hausvogteiplatz 5–7, Berlin, 1086; Tel. 2077530

Dr. Pieper/Schönberg

Zwei Routinen für KC 85/3

In MP 5/1989 wurde in der Rubrik Börse ein Programm für „80 Zeichen pro Zeile für KC 85/3" vorgestellt. Ein ähnliches Programm habe ich 1988 auch geschrieben. Es bietet nicht die Ausgabe von 80 Zeichen über PRINT#3 o. ä. sondern die im normalen Modus. Sowohl im Betriebssystem als auch in Basic ermöglicht es ein effektives Arbeiten. Zur Arbeit im Betriebssystem mußte der Bildschirm in 2 × 40 Zeichen pro Zeile geteilt werden. In Basic ist diese Teilung auch nutzbar. Weiter können über einen VPOKE Fenster (mit bis zu 32 Zeilen × 80 Spalten) festgelegt werden. Zusätzlich wurde eine neue Eingaberoutine entwickelt. Die Taste SHIFTLOCK wurde zur CTRL-Taste umfunktioniert, so daß alle Steuerzeichen (01H – 1AH) – diese wurden um einige Funktionen erweitert – von der Tastatur aus aufgerufen werden können. Es existieren noch einige Vorteile mehr (Zeichensatz, Blockgrafik, Tastaturbelegung, ...). Das Programm wurde in Assembler geschrieben und kann somit auf mehreren Speicherbereichen laufen (mit geringen Änderungen). Es belegt einen Speicher von etwa 2 KByte (je nach Version).
Das Programm und eine ausführliche Dokumentation stehen Interessenten unentgeltlich zur Verfügung.

Steffen Albrecht, Alte Straße 18, Herda, 5901

RISC-Chip mit 33 MHz

Von dem nach eigenen Angaben höchstintegrierten RISC-Prozessor, der derzeit zur Verfügung steht, liefert die Firma Motorola jetzt erste Muster aus. Es handelt sich dabei um eine 33-MHz-Variante des seit 1988 hergestellten Motorola 88000. Cache, Cachecontroller, Integereinheit und Gleitkommaeinheit sind auf zwei Bausteinen bereits enthalten; gefertigt wird der Chip in 1,2-μm-HCMOS-Technologie, die ab 1990 in die 1-μm-Technologie übergeleitet werden soll. Die Architektur bietet gute Voraussetzungen für Fehlertoleranz und Multiprocessing. Als Leistung werden 28 MIPS angegeben, die noch 1990/1991 auf 50 bis 90 MIPS gesteigert werden sollen. *MP*

Steigerung der Chipproduktion in Ungarn geplant

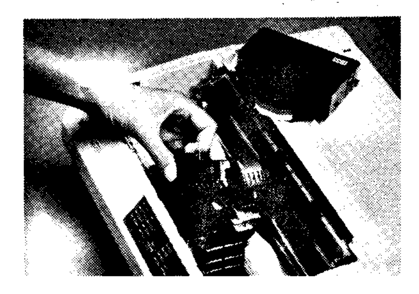
In sowjetisch-ungarischer Gemeinschaftsarbeit soll in Ungarn bis 1992 eine Halbleiterfabrik aufgebaut werden, die im ersten Produktionsjahr 15 Millionen integrierte Schaltkreise herstellt. Zu diesem Zweck wurde ein sowjetisch-ungarisches Joint-Venture-Unternehmen, Intermos, gegründet. Wie die Zeitschrift „eee" berichtete, soll „durch den Einstieg in die MOS-Technik der technologische Rückstand von zehn auf zwei bis drei Jahre verringert" werden. *MP*

80x86-Reihe ohne Ende?

Noch tun sich auf dem internationalen Markt Besitzer von PC ATs (mit dem 16-Bit-Prozessor Intel 80286) schwer, auf die leistungsfähigeren Nachfolger mit dem 32-Bit-Prozessor 80386 umzusteigen. Noch kämpft man mit Kompatibilitätsproblemen, die der neueste Prozessor, der 80486 aufwirft – da machen sich Intels Entwerfer bereits Gedanken um den Prozessor des Jahres 2000. Wir hatten ja bereits berichtet, daß noch schnellere Varianten des 80486 vorbereitet werden. Ende 1990 soll beispielsweise eine 50-MHz-Version zu erwarten sein, die PCs mit einer Leistung von 50 VAX-MIPS zuläßt (bei der Angabe VAX-MIPS wird die Leistung des von der Firma DEC 1977 auf den Markt gebrachten ersten, „klassischen" 32-Bit-Minirechners VAX 11/780 als Vergleichsmaßstab zugrunde gelegt – etwa 1 Million Instruktionen pro Sekunde); die geplante ECL-Variante des 80486 soll dann ab 1992 bei 25 MHz Taktfrequenz bereits 100 MIPS erreichen. Die Planung des Nachfolgers 80586, eines echten 64-Bit-Prozessors, ist bereits abgeschlossen, und erwartet wird dieser Prozessor in 3 bis 4 Jahren. Nun wurden bereits die ersten vagen Angaben zu den weiteren Entwicklungen gemacht. Als Erscheinungsdatum für den 80686 wurde das Jahr 1996 genannt; auf dem Chip sollen 22 Millionen Transistorfunktionen integriert werden. Der Mikroprozessor für das nächste Jahrtausend soll dann der 80786 werden: Auf etwa 6,5 Quadratzentimeter Fläche plant man, 100 Millionen Transistoren unterzubringen. Bei einer Taktfrequenz von 250 MHz werden Leistungen von einigen Milliarden MIPS erwartet. Auslieferung dieses Mikroprozessors soll das Jahr 2000 sein.
Auch wenn eine solche Ankündigungspolitik manchem unseriös erscheinen mag, sie läßt doch die enormen Anstrengungen in der Entwicklung der Mikroelektronik/Computertechnik deutlich werden. Wie realistisch die Aussagen letztlich sind bzw. waren, werden Sie spätestens in MP 1/2001 lesen können. *MP*

Umrüsten zum Farbdrucker

Für ihre Matrixdrucker der Reihe XB und FR bietet die Firma Star Micronics einen Aufrüstsatz an (s. Bild unten links), mit dem die Schwarzweißdrucker in Farbdrucker verwandelt werden können. Von der Firma wird neben der einfachen Handhabung – die Umrüstung ist so einfach wie das Auswechseln einer Farbbandkassette und kann deshalb von jedem Kunden selbst vorgenommen werden – vor allem der niedrige Preis des Aufrüstsatzes von 98 DM hervorgehoben. *MP*

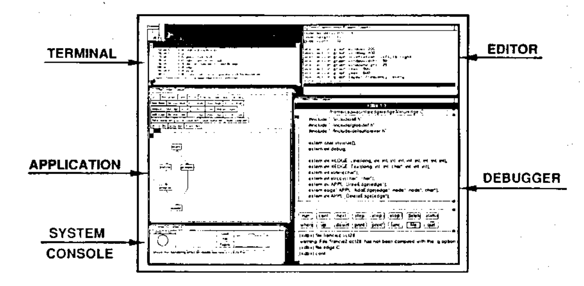


Hefen als lebende Halbleiterfabriken?

Jüngsten biotechnischen Forschungen an der Universität von Utah in Salt Lake City zufolge könnten Hefen in Zukunft auch als Produzenten für bestimmte Halbleitermaterialien genutzt werden. Die USA-Forscher machten zwei Hefe-Arten, Candida glabrata und Schizosaccharamyces pombe ausfindig, die in der Lage sind, einzelne Cadmiumsulfid-Kristalle zu bilden und abzuscheiden. In der Fachwelt gilt diese Entdeckung vor allem deshalb als bemerkenswert, weil die Herstellung dieses Halbleiterwerkstoffs für lichtempfindliche Elektronik-Sensoren und spezielle Katalysatoren in der Chemie bislang ziemlich kostspielig ist. Aufwendige Schutzmaßnahmen wegen der Giftigkeit des Cadmiums sind außerdem erforderlich.
Die biotechnische Möglichkeit geht nun von Peptiden (Aminosäurenverbindungen) und leicht handhabbaren Cadmiumsalzen aus. Der Kristallisationsprozeß selbst verläuft dann in zwei Stufen. Die Peptide bilden zunächst eine Art Matrix, an der die Hefen Reaktionen mit dem Cadmiumsalz hervorrufen, ohne daß es zu Vergiftungen der Hefezellen kommt, denn Cadmiumsalze gelten normalerweise als Zellmembrangifte. Die entstehenden Kristalle sind ausreichend groß, um sie als Sensorwerkstoffe einsetzen zu können. Bis ins Detail ist die biologische Reaktionskette jedoch noch nicht geklärt. Daher bleibt auch abzuwarten, ob sich das Verfahren großtechnisch nutzen läßt. Mikrobiologen halten es für durchaus denkbar, größere Kristalle schneller dadurch zu erzeugen, daß das Enzymsystem der Hefen isoliert und vom Organismus getrennt genutzt wird. *ADN*

Vereinfachte Benutzeroberfläche für PCs und Workstations

In den Siemens-Forschungs-Laboratorien in Princeton, New Jersey, wurde eine neue Benutzeroberfläche für PCs und Workstations entwickelt. Im Gegensatz zu den bisherigen Verfahren werden die zu verschiedenen Aufgabenbereichen gehörenden Fenster (Windows) auf dem Bildschirm nicht überlappend, sondern in verschiedener Größe nebeneinander angeordnet (s. Bild unten, rechts). Da auf diese Weise die Information aller Teilaufgaben sichtbar bleibt, wird die Arbeit bei komplexen Anwendungen erleichtert, besonders wenn mehrere Unterprogramme gleichzeitig bearbeitet werden müssen.



Bei den üblichen Window-Managern wird die Information auf dem Bildschirm in einzelnen Fenstern angeboten, die wie Papierseiten aufeinandergelegt werden. Soll ein Eintrag auf einer weiter unten liegenden Seite geändert werden, müssen die darüberliegenden Fenster verschoben oder geschlossen werden. Erhältlich sind heute auch schon Window-Manager, bei denen die Fenster nebeneinander in Spalten vorgegebener Breite angeordnet werden. Dies führt zu einer den Aufgaben nicht entsprechenden Aufteilung der Bildschirmoberfläche. Weniger wichtige Informationen erhalten zuviel Raum – Grafiken dagegen werden zu klein dargestellt.
Das von Siemens entwickelte Verfahren arrangiert die Lage und Größe der Fenster selbsttätig. In Anlehnung an das Anbringen von Kacheln (Tiles) verschiedener Größe, Form und Farbe an einer Wand wird das neue Verfahren als *Tiled Window-Technik* bezeichnet. *MP*

Hochgeschwindigkeits-Chips für RISC-Systeme

Erstmals seit dem Einstieg in die RISC-Technologie 1986 vergab Hewlett-Packard Mitte dieses Jahres Lizenzen auf die HP Precision Architecture (HP PA). Gemeinsam mit Hitachi startet HP die Entwicklung eines neuen Hochgeschwindigkeits-Chipsatzes. Auf der Basis von Hitachis Erfahrung als Halbleiterhersteller und der HP PA-Technologie sollen Chips für hochleistungsfähige RISC-gestützte Computersysteme entstehen, deren Software auf OSF-Erfordernisse abgestimmt ist.
Samsung, einer der großen Halbleiterhersteller der Welt, will gemeinsam mit HP neue, leistungsfähige RISC-Chipsätze als Grundlage für Workstation-Einstiegsmodelle mit HP PA entwickeln. Die Chips werden von Samsung in HPs CMOS-Technologie produziert. Das koreanische Unternehmen soll HP mit den Chipsätzen beliefern. Darüber hinaus darf Samsung die Chips in eigenen Workstation-Produkten einsetzen und auch Dritten anbieten. Die Vermarktung der Workstations selbst erfolgt getrennt.
Samsung wird seine Workstations mit HP-UX, der HP-eigenen Version des Betriebssystems UNIX, ausrüsten. Zukünftig soll Software der OSF verwendet werden. Neben einer Ausweitung der HP PA-Rechnerfamilie strebt HP die Position als führender Anbieter RISC-gestützter Systeme an. *MP*

Lisp für Echtzeitapplikationen

Für die von der IBM Japan Ltd. entwickelte Parallelprozessor-Workstation Top-1 hat die Firma jetzt eine neue Programmiersprache für Projekte der Künstlichen Intelligenz entwickelt. *Top-1 Common Lisp* hat eine Lisp-Implementierung der Universität Kyoto zur Grundlage und wurde mit Erweiterungen speziell für den Echtzeitrechner Top-1 versehen. Da der Rechner unter anderem auch zur Robotersteuerung eingesetzt werden soll, spielt das Problem der Garbage collection (Müllbeseitigung) eine wesentliche Rolle. Dabei werden überflüssige Daten gelöscht, sobald der Speicherplatz knapp wird; mit der Folge, daß der Rechner einen, wenn auch nur kurzen, Zeitraum seine Echtzeitaufgaben nicht erfüllen kann. Mittels eines neuen Speicherschemas soll dieses Problem bei der Lisp-Version von IBM Japan gelöst worden sein. *MP*

Neues Verfahren zur Herstellung von Halbleiterschichten

Ein neuartiges Verfahren zum Herstellen ultradünner Halbleiterschichten wurde von Wissenschaftlern der Universität Texas vorgestellt. Das als *Remote Plasma Enhanced Chemical Vapour Deposition* bezeichnete Verfahren arbeitet mit Temperaturen um 150°C, und nicht wie das heute übliche Verfahren mit Abscheidungen aus der Dampfphase, bei dem um 300 bis 600 Grad höhere Temperaturen benötigt werden. Der Vorteil des neuen Verfahrens besteht darin, daß auch Halbleiterwerkstoffe als Ausgangsmaterial für Chips verwendet werden können, die bisher wegen ihrer Temperaturempfindlichkeit ungeeignet waren. Erste Versuchsergebnisse sollen Chips ergeben haben, die anstelle von Millionen Transistorfunktionen etwa eine Milliarde auf der gleichen Fläche unterbringen.
Die industrielle Nutzung soll erst in knapp 10 Jahren möglich sein, da die komplizierte Steuer- und Regelungstechnik völlig neu entwickelt werden muß und alle verwendbaren Halbleitermaterialien auf ihre Eignung untersucht werden sollen.

Quelle: Blick durch die Wirtschaft vom 26. 6. 1989 Wi

Drucker mit zwei Druckköpfen

Um die Vorteile von Matrixdruckern und Typenraddruckern zu vereinen, hat die Firma Brother einen Drucker entwickelt, der sowohl über einen Matrixdruckkopf als auch über ein Typenrad verfügt. Beide sind auf einen Druckwagen montiert, somit können beide auf demselben Dokument und in derselben Druckzeile schreiben. Beispielsweise kann ein Text mit dem Typenrad geschrieben und die Grafik mit dem Matrixkopf gedruckt werden. Der neue Drucker stellt hohe Anforderungen an die Software, da die Programme die Befehle der Druckkopfansteuerung enthalten müssen. Trotz der Ankündigung der vollen Kompatibilität zu bestehenden Programmen ist Programmieraufwand notwendig, wenn die Möglichkeiten des doppelten Druckkopfes genutzt werden sollen.

Quelle: Frankfurter Allgemeine vom 4. 7. 1989 Wi

Farb-Flachbildschirm für tragbare Computer

Ingenieure der Firma IBM sollen an der Entwicklung eines Farb-Flachbildschirms arbeiten, der aus einer Kombination der Halbleitertechnik mit Flüssigkristallverfahren besteht. Die neue Technik ermöglicht eine Bildschirmmatrix mit 1,5 Millionen Farbbildelementen bei einer Bildschirmdiagonale von 36 cm. Die Tiefe des Bildschirms beträgt 3,8 cm. Gegenwärtig ist die Darstellung mit 16 Farben möglich. Das Kontrastverhältnis liegt mit 20 zu 1 sehr hoch, und durch die Beleuchtung von hinten kann der Schirm auch bei ungünstigen Beleuchtungsverhältnissen genutzt werden.

Quelle: Eildienst. – Berlin (1989) 147. – S. 4 Fa

Wettlauf bei der Produktion von Superchips

Den Markt für Superchips sollen gegenwärtig zu 90 Prozent japanische Firmen beherrschen. Als Konkurrent werden auch die westeuropäischen Firmen angesehen, die im Rahmen eines von der Europäischen Gemeinschaft finanzierten Projektes Anschluß an den Weltstand finden wollen. Wichtigste Partner sind Philips, SGS-Thomson und Siemens. Das Forschungsprogramm der EG wird mit 600 Mio Dollar beziffert.
Der gemeinsam von Siemens und Philips entwickelte 4-MBit-Chip soll auf SRAM-Basis (Static Random Access Memory) beruhen. Obwohl SRAM und DRAM ähnliche Anwendungsgebiete besitzen, ist die statische Variante zwar komplizierter im Herstellungsprozeß, jedoch schneller. Bei Philips soll Ende dieses Jahres die Entscheidung fallen, ob eine Chipfabrik für 2,3 Mrd. Gulden gebaut wird.
Zum Zeitpunkt der Gründung der Firma U. S. Memories – wir berichteten darüber in MP 9/89, S. 284 – wurde von IBM, einem Initiator dieser Gründung, die Produktion von 4-MBit-Chips aufgenommen. Die Aufnahme der Produktion des 1-MBit-Chips, auf der Basis japanischer Lizenzen, und des 4-MBit-Chips soll Entwicklungskosten von 3 Mrd. DM erfordert haben. Die Investitionen im Werk Böblingen werden auf 500 Mio DM geschätzt. Parallel dazu soll der 4-MBit-Speicher in den USA und später im IBM-Werk Yasu in Japan produziert werden. Der 1-MBit-Chip wird noch ein bis zwei Jahre gefertigt werden. Mit Beginn der 4-MBit-Chip-Produktion ging IBM als erster Produzent auf eine Siliziumscheibe mit 200 mm Durchmesser über (1-MBit-Chip: 135 mm). Dadurch kann die Produktivität wesentlich erhöht werden. Diese Steigerung wurde durch folgende Neuerungsschritte möglich:

– Eine weiterentwickelte CMOS-Technologie, bei der sowohl der Stromverbrauch als auch die Wärmeentwicklung im Chip während des Betriebes deutlich geringer sind, erlaubt eine höhere Packungsdichte.
– Einbeziehen der dritten Dimension beim Chip-Design. Die Verwendung einer platzsparenden Grabenstruktur für den Speicherkondensator ist mit einer dichteren und zuverlässigeren Verdrahtungstechnik verbunden.
– Die verbesserten fotolithografischen Prozeßtechniken erlauben eine Erhöhung der Justiergenauigkeit der Fotomasken beim Belichten der Wafer.

Im Gegensatz zur allgemein verbreiteten Auffassung von der Überlegenheit japanischer Chiphersteller ist IBM der Auffassung, daß sie mit Beginn der Massenproduktion (Millionen Chips/Monat) einen Vorsprung gegenüber japanischen Produzenten besitzt. Innerhalb der nächsten Monate sollen die 4-MBit-Chips in IBM-Produkten bereits wirksam werden.
Die bei 1-MBit-DRAMs führende japanische Firma Toshiba liefert z. Z. „mehrere Zehntausend" 4-MBit-DRAMs/Monat aus. Mitte 1990 sollen 1 Mio Stück/Monat ausgeliefert werden. Hitachi produziert 10 000 Stück/Monat. Ende 1989 sollen durch Inbetriebnahme einer 2. Fertigungslinie 400 000 Stück/Monat produziert werden. NEC produziert Handelsmuster im Labor, ab März 1990 soll die erste Fertigungslinie mit einem Ausstoß von 200 000 Stück/Monat die Produktion aufnehmen. Matsushita baut Fertigungslinien und will die Produktion im April 1990 aufnehmen. Mitsubishi und Sharp wollen die Serienproduktion Ende 1989 beginnen. Sanyo plant die Auslieferung von Handelsmustern für April 1990. Fujitsu liefert Muster aus. Angaben zur Serienproduktion wurden noch nicht veröffentlicht.

Quellen: Electronics Weekly. – London (1989) 1459. – S. 6
Neue Zürcher Zeitung vom Juni 1989
Handelsblatt vom Juni und Juli 1989
Der Tagesspiegel vom Juni 1989 Wi

Elektronisches Speicher-Subsystem auch für Großrechner

Um ein elektronisches Speicher-Subsystem auf Halbleiterbasis erweitert die Bull-Gruppe ihr Peripherieangebot für Mainframe-Anwender. Das System mit der Bezeichnung **RSS 8500** (RAM Storage Subsystem) bietet einen hohen Ein-/Ausgabedurchsatz mit Schnellzugriff auf die gespeicherten Daten.
Die neuartige Speicherperipherie, die auch als electronic disc bezeichnet wird, ergänzt das aktuelle Plattenspeicherangebot von Bull. Da der mechanische Zugriff entfällt, lassen sich gegenüber herkömmlichen Platten die Zugriffszeiten auf bis ein Zehntel reduzieren. Somit können bei Batch- und Transaktionsanwendungen die Gesamtdurchsatzraten wesentlich gesteigert bzw. die Antwortzeiten an den Terminals deutlich verkürzt werden. Je nach Ausstattung hat das neue elektronische Speicher-Subsystem eine Zugriffsgeschwindigkeit von 2 bis 5 Millisekunden (RAM-Zugriffszeit: 0,3 Millisekunden) und eine Speicherkapazität von 128 bis 4096 MByte (4 Gigabyte).
Die Kompatibilität zu vorhanden Magnetplattenanwendungen ist durch die Einteilung des RSS 8500 in logische Platteneinheiten gewährleistet.
Bei einem eventuellen Stromausfall sorgen integrierte Plattenlaufwerke und Batterien automatisch für die Datensicherung, um bei Wiederanlauf die erstellten Sicherheitskopien in den RAM-Speicher zu laden. MP

Trends bei der Gehäusefertigung

Höhere Integrationsdichte, größere Anforderungen an die Klimatisierung, ein absatzförderndes Design und rationelle Montageprozesse sind die Forderungen, die an die zum integralen Bestandteil der modernen Elektronik gewordene Gehäusetechnik gestellt werden.
Die Gehäusehersteller bieten, neben einem aus modernen Elementen aufgebauten Gehäusesortiment, zunehmend kundenspezifische Gehäuse an. Um auf kundenspezifische Wünsche reagieren zu können, die nicht durch ein Standardprogramm abgedeckt werden können, verfügen die Gehäusehersteller über CAD-Anlagen und CNC-Bearbeitungsmaschinen.
Die Montagefreundlichkeit der Gehäuse wird durch Schnappverschlüsse bei Halbschalenverbindungen, Schnellverschlüsse für Schrauben, vorgelochte Montageplatten und Steckmontagehilfen erhöht.
Unterschiedlich sind die Auffassungen zur Schraubverbindung. Obwohl einige Firmen ohne Schraubverbindungen auskommen und Schnapp-Rast-Technik einsetzen, wird bei nachträglichen Veränderungen der Einbaubedingungen die Schraubverbindung als vorteilhafter angesehen. Bei hohen Sicherheitsbedingungen und hoher Stabilität bleibt die Schraubverbindung die günstigere Verbindung.
Die Tendenz in Richtung Kunststoff als Gehäusematerial hält weiter an. Dabei erhöhen sich die Anforderungen an das Gehäuse hinsichtlich ihrer Schutzfunktion, bezogen auf Wärmeableitung, HF-Schirmung und höhere Schutzarten (IP 65).
Die Bedeutung des Designs nimmt aus Gesichtspunkten des Absatzes immer mehr zu.
Gründe für neue Anforderungen an Gehäuse:

– Die hohe Integration der elektronischen Bauelemente in Verbindung mit einer hohen Packungsdichte, niedrigem Leistungsniveau und hohen Taktfrequenzen führt zu einer hohen Beeinflußbarkeit der Systeme durch elektrische oder magnetische Störeinwirkungen.
– Wegen der höher werdenden Integration der Komponenten werden die Systeme in ihren Gesamtabmessungen kleiner. Die Systeme, die früher in einer Schrankreihe untergebracht waren, beanspruchen heute nur noch einen Schrank. Das, was früher in einem 19-Zoll-Schrank integriert war, findet heute Platz in einem Kleinschrank oder Tischgehäuse. Diese Entwicklung hält weiter an.
– Die Unterbringung von Bussystemen und Massenspeichern (Floppy, Winchester, Streamer) erfordert eine höhere Flexibilität im Innenaufbau und in der Anschlußtechnik.
– Um die Mensch-Maschine-Kommunikation zu verbessern, werden Ein- und Ausgabegeräte integriert (Tastatur, Bildschirm usw.).
– Die immer höheren Packungsdichten der Elektronik lassen eine größere Wärmeproblematik erwarten.
– Der zunehmende Einsatz der Elektronik in der Industrie erfordert wegen der gesteigerten industriellen Umweltforderungen die Integration einer Vielzahl von peripheren Geräten.
– An das Design werden immer höhere Anforderungen gestellt. Die Gehäuse müssen sich in eine Bürolandschaft genauso einfügen wie in die Einsatzgebiete der Regel-, Steuer- und Meßtechnik.
– Auch die Daten- und Zugriffssicherheit gewinnt immer mehr an Bedeutung.
– Nicht zuletzt bedeutet die Verbesserung des Preis/Leistung-Verhältnisses in der Elektronik auch eine Herausforderung für die Mechanik.

Quellen: Produktion. – Landsberg (1989) 28. – S. 10, 11
Elektronik. – München 38 (1989) 13. – S. 130 Wi

89. Budapester Internationale Messe

Vom 17. bis 25. Mai 1989 fand in Budapest die 89. Internationale Messe als Fachmesse für Investitionsgüter statt. Mit 1861 lag die Zahl der Aussteller um knapp 200 unter der des Vorjahres. So machte sich deutlich das fast völlige Fehlen von Computeranbietern aus Taiwan bemerkbar (ebenso wie übrigens auch auf der Leipziger Frühjahrsmesse). Trotzdem war zu erkennen, daß der Elektronikbereich als „Motor der Spitzentechnologie" favorisiert wird. Nach Meinung der Veranstalter zeigte die Messe insgesamt „trotz der schwierigen Lage der ungarischen Wirtschaft als eine Art zuversichtlich stimmendes Barometer jene Veränderungen an, die der Wirtschaft helfen können, den toten Punkt zu überwinden". Äußeres Zeichen für diese Veränderungen war, wie im Vorjahresbericht schon bemerkt, die relativ große Zahl von Betrieben verschiedener Eigentumsformen, die zunehmend mit ausländischem Kapitalanteil gegründet werden. Viele haben den Charakter von Montagebetrieben, die lediglich importierte Baugruppen montieren, oder Zwischenhändlern. Oft heben Softwareanbieter hervor, Lizenznehmer und damit offizieller Distributor für international bewährte Standardsoftware zu sein. Jedoch gibt es auch viele, die sich auf das Erstellen bzw. Anpassen von Anwendungssoftware entsprechend den Kundenwünschen spezialisiert haben. Die meisten dieser Firmen haben eine breite Palette zu bieten, wie Farbbild 1 zeigt. An zahlreichen PC-Arbeitsplätzen demonstrierte zum Beispiel die Firma Controll die Anwendung vor allem von CAD-Software.

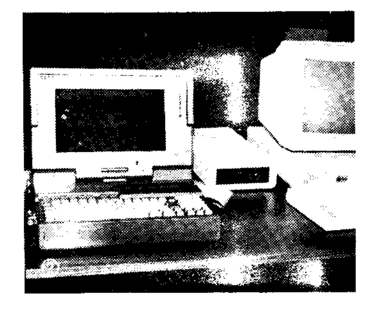
Anbieter ähnlicher Art waren azsio, Softinvest, Procontrol, Lezer u. a. Die Mehrzahl konzentriert sich dabei auf die Personalcomputertechnik bzw. deren Anwendungen wie CAD und DTP, wobei auch Systeme mit 32-Bit-Prozessoren nicht ausgeschlossen sind. Ebenfalls im Trend liegen transportable Personalcomputer, vor allem in der Laptop-Klasse. Originell, und von den Laptopherstellern bisher wohl noch nicht propagiert, ist ein Einsatzgebiet, das die ungarischen Czepel-Werke für diese Technik erschlossen haben. Sie schufen eine Lösung, mittels der ein Toshiba-Laptop zur Programmierung von NC-Werkzeugmaschinen verwendet wird. Der Bediener schließt dazu den Laptop an die jeweilige Maschine an, der die Steuerdaten nunmehr über die serielle Schnittstelle übermittelt werden können (Farbbild 2). Das Erstellen der Programme ist auf einem PC AT-kompatiblen Arbeitsplatz möglich, der natürlich prinzipiell auch selbst mit der Maschine – über RS-232-Schnittstelle – gekoppelt werden kann.

Als Tochterfirma von Elektromodul wurde während der Messe speziell für den Vertrieb von Computertechnik die Kooperative Digitmodul gegründet. In ihrem Angebot befand sich unter anderem ein 32-Bit-PC in Towerform. Der **HAT-31/3** (Bild 1) hat den Prozessor Intel 80386 in der 16-MHz-Version. Der RAM ist bis auf 8 MByte ausbaufähig und umfaßt auch die Shadow-RAM-Funktion. Die drei lieferbaren Varianten des Computers unterscheiden sich in der Festplattenausstattung. 159,8 MByte bei 28 ms mittlerer Zugriffszeit bietet das Modell HAT-31/3160 in der größten Ausbaustufe. Weitere Massenspeicher sind ein 5,25-Zoll-Floppylaufwerk mit 360 KByte oder 1,2 MByte und ein 3,5-Zoll-Floppylaufwerk mit 720 KByte oder 1,44 MByte. In dem Gehäuse ist Platz für zusätzliche 5 AT- und 2 XT-Steckkarten. Zur Ausstattung gehören eine parallele und zwei serielle Schnittstellen sowie verschiedene Monitore. Die ausgestellte Konfiguration beinhaltete einen Multisync-Monitor mit VGA-Grafik.

Nach eigenen Aussagen noch immer expandierend ist die im Vorjahresbericht bereits erwähnte Genossenschaft Müszertechnika. Hervorgehoben wurde in diesem Jahr die Präsentation von Microsofts OS/2-LAN-Manager. Aus der Angebotsvielfalt an Hardware sei hier nur der transportable PC **MAT/H2** genannt, der mit einem Elektrolumineszenz-(EL-)Display ausgestattet ist (Farbbild 3). Er besitzt einen 16-MHz-80286-Mikroprozessor (auch Einsatz des 80386 mit 20 MHz möglich), wobei die CPU nicht auf einem Motherboard, sondern wie die anderen Bauelemente auf einer Steckkarte angeordnet ist. Von den 5 Steckplätzen des Gerätes bleiben nach Abzug des CPU-Steckplatzes und eines Steckplatzes für den Monitor noch drei Steckplätze zur freien Nutzung für den Anwender. In das Gehäuse sind zwei 3,5-Zoll-Floppylaufwerke zu 1,44 MByte und eine Harddisk mit 27 MByte integriert.

Die Firma Makrotrend dürfte wie Müszertechnika einer der wenigen Aussteller gewesen sein, der seine Produkte selbst entwickelt und fertigt. Spezialisiert ist das Unternehmen auf Produkte zur Vernetzung von PCs und hat nach eigenen Angaben 60 Prozent des Marktanteils von Netzelementen in Ungarn. Das Farbbild 4 zeigt eine ARCNET-kompatible **LAN-Interfacekarte.** Sie ist einsetzbar in PC XT/AT- und kompatible Computer und entspricht der Novell Advanced Netware 86/286. Die maximale Übertragungsrate beträgt 2,5 MBit/s.

Von der Möglichkeit, mit ausländischen Firmen gemischte Unternehmen gründen zu können, macht seit Anfang des Jahres auch Ungarns größter Computerproduzent, Videoton, Gebrauch. Ziel ist es, durch dabei erworbenes Know-how konkurrenzfähigere Produkte anbieten zu können. So wurden – unter dem Dach von Videoton-Elektronik als Holdinggesellschaft – in letzter Zeit zahlreiche relativ eigenständige Firmen gegründet. Auf dem weitläufigen Stand Videotons konnte die Zuordnung der Exponate zum Hersteller da schon Schwierigkeiten bereiten (siehe Farbbild 5). Die Walton GmbH, deren Standbereich das Bild zeigt, ist eine Verbindung mit der britischen Firma Walters International. Ausgestellt waren die folgenden Produkte. **Ein Laptop** (Bild 2), ausgestattet mit 80286-Prozessor, 640 KByte RAM, 1 × 40 MByte Festplatte, 1 × 3,5-Zoll-Floppy mit 1,44 MByte, einem LC-Display im EGA-Modus (640 × 350 Pixel) und 2 Slots für Erweiterungskarten (beispielsweise Ethernet-Karte).




③


④



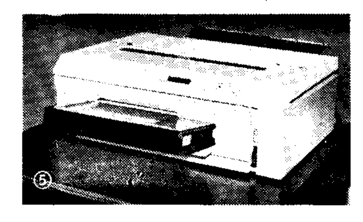
①

Als Einstieg in die 32-Bit-Klasse ist **der SX 386** anzusehen (Bild 3). Er arbeitet mit dem Prozessor 80386 SX, der zwar extern nur einen 16-Bit-Datenbus besitzt, dafür aber die verbreitete und billigere Peripherie der AT-Klasse nutzen kann. Wegen des internen 32-Bit-Busses ist jedoch die gesamte Softwarepalette des „echten" 32-Bit-Prozessors 80386 (neuerdings auch oft als 80386 DX bezeichnet) lauffähig (siehe auch MP 9/88, S. 278). Der 1-MByte-RAM kann bis auf 8 MByte ausgebaut werden. Neben dem 1,2-MByte-Floppylaufwerk lassen sich Festplatten von 40 oder 80 MByte nutzen.

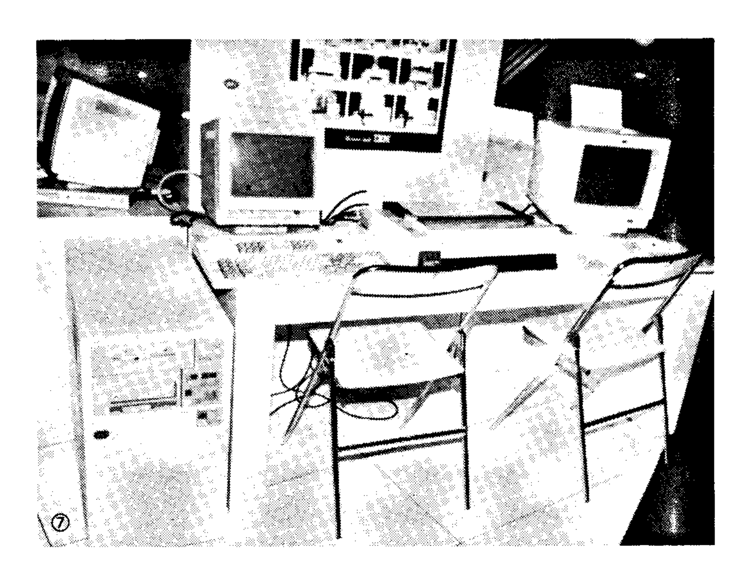
Das leistungsstärkste Modell war der **Baby 386 AT** (Bild 4), ausgerüstet mit einem 16-MHz-Prozessor 80386. Der 2-MByte-RAM läßt sich auf der Hauptplatine bis zu 4 MByte aufrüsten. Neben dem 5,25-Zoll-Floppylaufwerk mit 1,2 MByte ist eine 40-MByte-Festplatte vorhanden. Monitore stehen in verschiedenen Ausführungen zur Auswahl, gezeigt wurde der VGA-Modus in einer Auflösung von 800 × 600 Pixel.

Als bemerkenswert ist festzuhalten, daß alle genannten Walton-Computer über freie Steckplätze für LAN-Steckkarten verfügen und deren Vernetzung – einschließlich des Laptops – demonstriert wurde.

Von den zahlreichen anderen Erzeugnissen, die unter dem „Videoton-Dach" angeboten wurden, seien abschließend die Farbmonitorfamilie **Chameleon** sowie der Tintenstrahldrucker **SI 480** (Bild 5) genannt. Das Besondere an dem Druckprinzip des SI 480 ist, daß anstelle der sonst üblichen flüssigen Tinte Festtintenpellets verwendet werden. Die schwarze Tinte wird erst im Druckkopf verflüssigt und zum Druck mittels eines piezoelektrischen Systems auf das Papier geschleudert. Dort erstarrt sie innerhalb von 250 Millisekunden zu dokumentenechter Schrift.
Der SI 480 liefert im Grafikmodus eine Auflösung von 240 × 240 dpi (Punkte pro Zoll). Zeichen werden in Letter Quality (240 × 240 dpi) mit bis zu 680 Zeichen/s ausgedruckt, in Executive Quality (480 × 240 dpi) mit bis zu 340 Zeichen/s. Zum Darstellen verschiedener Schriften steht eine Vielzahl ladbarer Fonts (Zeichensätze) zur Verfügung, die in den 64-KByte-Fontspeicher geladen werden können.


⑤

Entsprechend dem internationalen Trend war das Angebot von Rechnern oberhalb der PC-Klasse relativ gering. So stellte lediglich das Kombinat Robotron neben seiner PC- und Schreibtechnik den Minicomputer K 1840 vor, und das Zentrale Forschungsinstitut für Physik der Ungarischen Akademie der Wissenschaften – KFKI – demonstrierte die **TPAstation TPA 11/530** als CAD-System (Bild 6). Im Gegensatz zu dem im Vorjahr gezeigten Modell TPA 11/510 erreicht die TPA 11/530 mit 2,7 MIPS (Millionen Instruktionen pro Sekunde) eine wesentlich höhere Leistung. Die in VLSI-Technik gestaltete CPU ist über den 32-Bit-LBUS mit dem 8 bis 64 MByte großen Hauptspeicher verbunden. Die Verbindung zu den peripheren Geräten erfolgt über den 16-Bit-QBUS. Hierüber wird auch die Vernetzung mit anderen Rechnern vorgenommen. So waren die auf dem Stand ausgestellten verschiedenen TPA-Rechner, aber auch XT- und AT-kompatible PCs, mittels Ethernet vernetzt. Beispielsweise der rechts im Bild zu sehende AT-kompatible Olivetti 28 mit der TPAstation. Auf letzterer wurde die Leiterkartenentwicklung demonstriert, auf dem Olivetti 28 ein Beispiel der mechanischen Konstruktion einschließlich des Erstellens der NC-Daten.
Als weiterer „Mittelklasse"-Rechner wurde das neue Anwendungssystem von IBM, **AS/400** (Bild 7), gezeigt. Die Firma will damit die sogenannten Midrange-Kunden ansprechen, für die es bisher die Modellreihen /36 und /38 gab. Das neue System besteht aus 6 Modellen, für die das Betriebssystem OS/400 entwickelt wurde, unter dem auch bisherige /36- und /38-Anwendungsprogramme laufen können. Herzstück der Rechner ist ein IBM-eigener 32-Bit-Hochleistungsprozessor. Während die kompakten Einstiegsmodelle B 10 und B 20 über einen maximalen Hauptspeicher von 8 bzw. 16 MByte verfügen und als Zentraleinheit bis zu 40 Bildschirme/Drucker „bedienen" können, reichen die Hauptspeicherkapazitäten der Standardmodelle B 30, B 40, B 50 und B 60 von 36 über 40 und 48 bis zu 96 MByte. Am Modell B 60 sind darüber hinaus Magnetplatten mit bis zu 8 GByte sowie bis zu 480 Datenstationen anschließbar. Nach Angaben von IBM liefert das AS/400 damit gegenüber dem System /36 die bis zu fünffache Leistung.
Von IBM war weiterhin der im Vorjahr herausgebrachte PC des Personal Systems, **PS/2 Mod. 50Z** (Farbbild 6) zu sehen. Das „Z" steht dabei für zero wait states; das heißt, aufgrund der neuerdings verwendeten DRAMs mit 85 Nanosekunden Zugriffszeit gegenüber bisher 120 ns braucht die mit 10 MHz getaktete 80286-CPU keine Wartezyklen einzulegen. Damit wird eine um 25 Prozent höhere Rechengeschwindigkeit erreicht. Weitere Änderungen betreffen die größere Festplatte – jetzt 60 MByte – und die Anwendung der SMD-Technik, womit sich die Größe der Hauptplatine um ein Drittel reduzieren ließ. Auch lassen sich auf der Hauptplatine jetzt 2 MByte gegenüber bisher 1 MByte RAM unterbringen, so daß auch bei Maximalausbau ein Steckplatz frei bleibt.
Zahlreiche weitere ausländische Hersteller hatten ebenfalls überwiegend PCs im Angebot, allerdings ohne ihre Spitzenprodukte. Das trifft insbesondere auf die österreichische Firma Theuretzbacher & Co. zu, deren mit dem 33-MHz-Mikroprozessor 80386 arbeitenden AT-386/33 wir in MP 8/88, 4. Umschlagseite, bereits vorstellten.


⑦

In Budapest wurde als leistungsstärkstes Gerät der Future Technologie **AT-286/12** in einer kompakten Desktopausführung gezeigt (Farbbild 7). Die Standardausstattung umfaßt u. a. 1 MByte RAM, ein 5,25-Zoll-Floppylaufwerk und Adapter für CGA- und MDA-Monitore. Als Optionen sind Festplatten bis 151 MByte (17 ms mittlere Zugriffszeit; ESDI-Schnittstelle) und Monitore für EGA, PGA und VGA verfügbar.
Auch bei der Firma ASEM reicht die Angebotspalette bis zu 386er Systemen mit 25-MHz-Takt und 0 wait states. Gezeigt wurde jedoch als leistungsfähigster PC das Towermodell **THOR 8070** (Farbbild 8). Es besitzt den 16-Bit-Prozessor 80286 mit 12 MHz Taktfrequenz und arbeitet ebenfalls ohne Wartezyklen des Prozessors. Optional ist der mathematische Koprozessor 80287. Der Standard-RAM von 1 MByte läßt sich bis zu 16 MByte erweitern. Das ausreichend dimensionierte Gehäuse bietet zwei XT- und sechs AT-Steckplätzen Raum. Massenspeicher sind ein 5,25-Zoll-Floppylaufwerk mit 360 KByte/1,2 MByte, ein 3,5-Zoll-Floppylaufwerk mit 720 KByte/1,44 MByte und eine Festplatte zwischen 20 und 320 MByte. Die Kapazität des Streaming-(Band-)laufwerkes beträgt zwischen 20 und 150 MByte. Als Betriebssystem sind MS-DOS, MS-OS/2 und Xenix 286 vorgesehen.
Aus dem vielseitigen Angebot der Budapester Messe lassen sich für den Bereich der Computertechnik vor allem folgende Trends ableiten:

- Die Mehrzahl der Anbieter konzentriert sich heute auf die sogenannte dezentrale Datentechnik. Minirechner werden vor allem als leistungsstarke Grafikarbeitsplätze oder für den Betrieb von zahlreichen Terminals eingesetzt (z. B. TPAstation und AS/400). Die dominierenden Personalcomputer basieren fast ausnahmslos auf den 16- und 32-Bit-Prozessoren von Intel, so daß durchgehende Kompatibilität gewährleistet ist (ausgenommen die Mikrokanalmodelle des PS/2 von IBM).
- Immer beliebter werden tragbare Computer, wobei die Klasse der „Schoßcomputer" – Laptops – den größten Anteil hat. Prozessorbasis sind durchweg Intelprozessoren und Kompatible.
- Der steigende Bedarf der Anwender, über den zunächst isoliert betriebenen „persönlichen" Computer mit anderen Abteilungen des Betriebes zu kommunizieren, hat zu einer Flut von Hilfsmitteln zur Vernetzung geführt. Im PC-Bereich dominieren dabei die Netze Ethernet und ARCNET. Hard- und Software dafür wird auch von zahlreichen ungarischen Firmen angeboten.

Text und Fotos: Hans Weiß


⑥

TERMINE

Tagung zu Wechselwirkungen zwischen Informatik und Kultur

WER? Gesellschaft für Informatik der DDR, Kulturbund der DDR und Kunsthalle Rostock
WANN? 19. bis 22. Dezember 1989
WO? Kunsthalle Rostock
WAS? Computer als Kulturgut; Ausstrahlung der Informatik auf das kulturelle Milieu; Beziehungen zwischen Rechentechnik und Kunst; kulturvoller Umgang mit der Rechentechnik; Nutzung des Computers in der Kultur; Ausstellung von Computergrafiken und Vorführung von Computermusik
WIE? Zur Erfassung der Interessenten wird um Anmeldung gebeten (Teilnahmegebühr ca. 100,– M). Anmeldungen oder Rückfragen richten Sie bitte an: Akademie der Wissenschaften der DDR, Zentralinstitut für Kybernetik und Informationsprozesse, Prof. Dr. Völz, Rudower Chaussee 5/6, Berlin, 1199; Tel. 6743141

Weichardt

Der 16-Bit-Mikroprozessor des ESER-PC

von Jochen Bonitz, VEB Verlag Technik, Berlin 1989, 1. Auflage, 204 S., DDR 20,– M

Anliegen des Buches ist die ausführliche Darstellung der Architektur, der Arbeitsweise und des Befehlskodes des Mikroprozessors K 1810 WM86, der zum i8086 der Firma Intel funktionell kompatibel ist. Dies sind notwendige Kenntnisse für Fachleute, welche den K 1810 WM86 einsetzen wollen.
In einem kurzen Einblick in die Entwicklungsgeschichte wird begründet, warum man gerade diese Architektur für den Prozessor gewählt hat. Einen guten Grundüberblick über die Hardwarearchitektur gibt das Kapitel 2 auf ca. 30 Seiten. Besonders wird auf jene Elemente eingegangen, die für das Verständnis zeitlicher Abläufe und für das Erfassen der Speicherorganisation wichtig sind. Ein eigenes Kapitel widmet der Autor den reichhaltigen Möglichkeiten, mit Maschinenbefehlen auf Operanden zuzugreifen. Diese Adressierungsarten sind wichtig für das Verständnis der Wirkungsweise der Maschinenbefehle. Dabei ist die Unterscheidung der Adressierungsart *relativ* und *indiziert* nicht durch die Architektur des Prozessors begründet, sondern durch die unterschiedliche Assemblernotation ein und desselben Maschinenbefehls. Dies wird in den Ausführungen nicht deutlich und könnte zur Verwirrung des Lesers führen. Eine Ungenauigkeit ist dem Verfasser bei den Beispielen zur indirekten Adressierung unterlaufen, wo er relative Adressierung/Offset 0 mit indirekter verwechselt, obwohl er zuvor richtig darstellt, mit welchen Registern indirekte Adressierung möglich ist.
In einer für die Einarbeitung in den Befehlssatz sehr zweckmäßigen Gliederung in logisch zusammengehörende Einheiten werden die Befehle des Prozessors auf 46 Seiten dargestellt. Der Leser gewinnt eine Vorstellung von der Mächtigkeit der Befehle und davon, was sie bewirken. Wenn er jedoch beginnt, selbst zu programmieren, wird er Angaben zu den möglichen Adressierungsarten in den Befehlen vermissen. Diesen ist ein eigenes Kapitel gewidmet. Es enthält auf weiteren 41 Seiten nochmals – diesmal alphabetisch – alle Befehle mit Kurzbeschreibung, Flagbeeinflussung, anwendbaren Adressierungsarten, Taktanzahl und binärer Kodierung. Gerade die Darstellung der möglichen Adressierungsarten ist sehr kompakt, was sicher vielen Lesern ein schnelles Erfassen der Varianten der Befehle erschwert. Die Trennung in allgemeine und umfassende Befehlsbeschreibung kommt sicher dem Leserkreis entgegen, welcher nur einen Einblick in die Möglichkeiten des Prozessors gewinnen möchte. Als Arbeitsmittel für den Praktiker erhöht sie jedoch den Aufwand für das Auffinden einer Information, der gesenkt werden könnte, wenn im 6seitigen Anhang „Befehle des K 1810 WM86" außer den wichtigsten Angaben auch Seitenverweise enthalten wären.
Der Abschnitt über die Assembler fällt dem Anliegen des Buches entsprechend kurz aus, vermittelt dem Leser aber einen guten Eindruck vom Zweck und der Funktion dieser Programmierwerkzeuge. Auf den verbleibenden 29 Seiten des Titels hat der Verfasser den erfreulichen Versuch unternommen, Aufbau, Arbeitsweise und Programmierung des Arithmetik-Koprozessors K 1810 WM87 einer breiteren Öffentlichkeit vorzustellen. Der Leser erhält Informationen, die es ihm ermöglichen, selbst einzuschätzen, bei welchen seiner Anwendungen er bei Einsatz des WM87 wesentlichen Effektivitätsgewinn erwarten kann. Eine allumfassende Beschreibung dieses Koprozessors kann aber nur eine eigenständige Schrift geben.
Alles in allem kann der vorliegende Titel neben Praktikern bei der Anwendung des WM86 vor allem jenen Lesern empfohlen werden, die gern mehr über das Herz ihrer heutigen 16-Bit-PCs erfahren möchten.

Dr. M. Krzikalla

„Neue Medien" – Kommunikation und Information

von J. Herzog u. M. Lare (Hrsg.), Lexikon, VEB Verlag Technik, Berlin 1989, 275 S., DDR 24,– M

Der Titel diese Lexikons bleibt bedenklich – und auch der Untertitel ist eher irreführend. Das, was das Lexikon enthält, sind Fachbegriffe der Nachrichten- und Computertechnik, vereinzelt sogar der Mikroelektronik und Robotertechnik. Eine Schrift über „Medien" müßte einen wesentlich anderen Inhalt haben. – Aber die vierte Umschlagseite hilft. So findet man im Buch viele moderne Begriffe, die man sucht – oder auch nicht sucht: Abonnementfernsehen, Bandbreite, CAD, DAC, ECMA, Fabry-Pérot-Resonator, GAA, Halbbildspuraufzeichnung, IBFN, Josephson-Computer, Kabelbild, LAN, MAC, Nachbarkanalleistung, Oberwellenfilter, PAD, Quadrofonie, Radio-Daten-System, Satellit, Tag, Überlagerungsempfang, VCR-Longplay-Verfahren, WAN, X.21-Schnittstelle und Zeichengenerator – Anfangsbegriffe der Abschnitte von A bis Z – ganz allgemein und auch ganz speziell.
Für den Leser der MP sind folgende Begriffe (u. a.) interessant: Biocomputer, Concurrent-PROLOG, Displayprozessor, Expertensystem, Inferenzmaschine, Frame, Projekte für Rechner der 5. Generation, RISC, SDL, Transputer. Das sind sehr wichtige und nützliche Dinge. Ihre Definitionen und Erklärungen sind allgemeinverständlich gehalten.
Das Lexikon enthält nur wenige Namen, die bedeutenden Repräsentanten der modernen Technik gehören. Erwähnung finden Czochralski, Josephson, von Neumann – als Beispiele. Ebenso wären noch mehr Zahlenangaben nützlich gewesen. Dagegen ist das Verzeichnis von Normen, Bestimmungen und Empfehlungen im Anhang zweifellos eine Rarität.
Das Lexikon ist für sehr viele Leser eine sehr wichtige Hilfe. Die behandelte Begriffswelt entspricht dem internationalen Gebrauch. Es ist zu hoffen, daß die Auflagenhöhe dem großen Interessentenkreis gerecht wird.

Prof. Dr. Dr. M. Roth

FORTRAN

von E. Stamer, Verlag Die Wirtschaft, Berlin 1988, 256 S., 10 Abb., ISBN 3-349-00411-3; DDR 16,– M

Das vorliegende handliche Taschenbuch schließt bei uns eine seit vielen Jahren klaffende Lücke in der Computerliteratur. Während frühere Bücher (z. B. Schoch/Schatz, Programmierung in FORTRAN, BSB B. G. Teubner Verlagsgesellschaft, Leipzig, 1971) ausschließlich für Spezialisten geschrieben waren, hat Stamer entsprechend dem aktuellen Trend zum massenhaften Einsatz von Arbeitsplatzrechnern ein durchaus anspruchsvolles Fachbuch in einem „sehr lockeren Stil" geschrieben, so daß sich auch jeder an Programmierung interessierte Leser schnell in die FORTRAN-Programmierung einarbeiten kann. Es ist verblüffend, wie einfach der Leser an die Sprachelemente von FORTRAN herangeführt wird. Das trifft vor allem auf die ersten vier Abschnitte zu. Die weiteren sieben Abschnitte stellen wesentlich höhere Anforderungen an den Leser und sollten nicht chronologisch und vor allem nicht ohne praktische Erfahrungen durchgearbeitet werden. So werden im fünften Abschnitt relativ komplizierte Ein- und Ausgabeprobleme am Beispiel einer etwas komplexeren Aufgabenstellung behandelt. Der sechste Abschnitt beinhaltet eine komplexere Aufgabenstellung unter Einbeziehung von Redabas und dem Textprozessor TP. Der siebente Abschnitt rundet durch weitere Sprachelemente das Buch für den Spezialisten ab, erhebt aber trotzdem keinen Anspruch auf vollständige Darstellung aller FORTRAN-Sprachelemente. Im achten Abschnitt stellt der Autor aus seiner Sicht eine Reihe praktischer Erfahrungen und Hinweise zusammen, die, obwohl manche Aussage vielleicht auch etwas umstritten ist, besonders für den Anfänger sehr nützlich sind.
Ein weiterer Vorteil des Buches ist seine konsequente Orientierung an FORTRAN-77, so wie es seit 2 Jahren gemäß TGL 44500 für die DDR verbindlich ist. Da es noch eine Reihe von FORTRAN-IV-Compilern für andere Rechner gibt, werden im neunten Abschnitt wesentliche Unterschiede zu FORTRAN-IV und anderen Compilern zusammengefaßt.
Der zehnte Abschnitt ist der Einbindung von Assemblerunterprogrammen gewidmet, während im letzten Abschnitt über die Programmierung hinausgehende Hinweise zur Softwareentwicklung gegeben werden.
Die Anhänge enthalten Hinweise zur Arbeit mit FORTRAN unter SCP auf A 7150 und PC 1715, eine Kurzfassung der TGL 44500 sowie die Lösungen zu den zahlreichen Übungsaufgaben, die man zur Selbstkontrolle unbedingt lösen sollte.
Trotz einiger kleiner Fehler, die sich bei der Druckaufbereitung der Beispiele eingeschlichen haben und den Gesamteindruck nicht schmälern, kann ich das Buch jedem an FORTRAN interessierten Leser nur empfehlen.

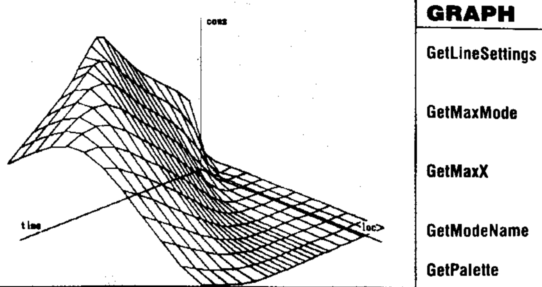
Prof. Dr. Th. Horn

CAD/CAM – Leitungsaufgaben und Erfahrungen

Autorenkollektiv, Verlag Die Wirtschaft, Berlin 1987, 160 S., 6,50 M

Zwölf Autoren des Zentralinstituts für sozialistische Wirtschaftsführung des ZK der SED, Berlin, legten unter Leitung der kompetenten Volkswirtschaftler H. Koziolek, W. Salecker und Th. Schmieder eine Schrift vor, deren hauptsächliches Ziel es ist, gestützt auf eine mehrjährige engagierte Arbeit zur Durchsetzung der Beschlüsse der Partei bezüglich der Schlüsseltechnologie CAD/CAM, verallgemeinerungsfähige Erkenntnisse und praktische Erfahrungen an einen möglichst großen Leserkreis zu vermitteln.
Allen Abschnitten der schnell lesbaren Schrift ist die Rechentechnik als modernes Arbeitsmittel gemeinsam, so wie sie sich in der DDR-Volkswirtschaft in weiter Verbreitung befindet. Die wichtigsten Aspekte der in der Reihe Blickpunkt Wirtschaft erschienenen Publikation sind: CAD/CAM als Schlüsseltechnologie, ihre Durchsetzung in der Volkswirtschaft, entsprechende Aufgaben der Leitungen, erforderliche Weiterbildung, Einsatzanforderungen, Organisation und Planung der effektiven Nutzung, beispielhafte Initiativen und vor allem gute Erfahrungen, die mit CAD/CAM-Systemen in Leitung und Verwaltung (Planung, Bilanzierung, Rechnungsführung, Statistik) gesammelt wurden. Immerhin macht dieser Einsatzbereich etwa 40 Prozent aller Rechneranwendungen aus. In einem kurzen Abschnitt (5.) werden „Grundbegriffe der modernen Rechentechnik in Verbindung mit der CAD/CAM-Technologie" behandelt. In den nicht sehr genauen Erläuterungen ist z. B. folgender Satz unter der Überschrift „Architekturprinzipien" von Rechnern völlig absurd: „Eine zunehmende Rolle in der weiteren technischen Entwicklung wird die Biotechnologie spielen".
Mit der Absicht, alle positiven und konstruktiven Aspekte zur Durchsetzung der CAD/CAM-Technologie in der DDR darzulegen, ist diese Schrift ein wichtiger propagandistischer Beitrag. Widersprüchliche Prozesse oder Hindernisse werden jedoch nicht analysiert, obwohl dies für einen so umfassenden und bedeutsamen Prozeß sehr förderlich wäre.

Prof. Dr. Dr. M. Roth



GRAPH

Routinen	
Arc	zeichnet einen Kreisbogen von einem Start- zu einem Endwinkel und verwendet eine Koordinate (x, y) als Kreismittelpunkt
Bar	zeichnet einen Balken mit der momentan gesetzten Farbe und dem aktiven Füll-Muster
Bar3D	zeichnet einen dreidimensionalen Balken mit der momentan gesetzten Farbe und dem aktiven Füll-Muster
Circle	zeichnet einen vollständigen Kreis mit gegebenem Radius um einen Mittelpunkt (x,y) herum
ClearDevice	setzt alle Parameter des Grafikpakets auf Standardwerte zurück
ClearViewPort	löscht das momentan gesetzte Grafikfenster
CloseGraph	beendet das Grafikpaket
DetectGraph	prüft die vorhandene Hardware und bestimmt, welcher Grafiktreiber geladen und welcher Modus gesetzt werden muß
DrawPoly	zeichnet einen Polygonzug in der momentan gesetzter Linienart und -farbe
Ellipse	zeichnet einen elliptischen Ausschnitt
FillEllipse	zeichnet eine ausgefüllte Ellipse
FillPoly	füllt ein Polygon
FloodFill	füllt einen umschlossenen Bereich mit dem momentan über SetFillStyle bzw. SetFillPattern gesetzten Muster
GetArcCoords	bestimmt die Koordinaten des zuletzt gezeichneten Kreisausschnitts
GetAspectRatio	bestimmt das tatsächliche Darstellungsverhältnis des benutzten Bildschirms
GetBkColor	liefert die momentan gesetzte Hintergrundfarbe
GetColor	liefert die momentan gesetzte Zeichenfarbe
GetDefaultPalette	liefert die Farbpalette, die bei der Initialisierung des Grafiktreibers aktiv war
GetDriverName	liefert den Namen des momentan aktiven Grafiktreibers als String
GetFillSettings	fragt die über SetFillStyle oder SetFillPattern gesetzten Parameter ab
GetGraphMode	liefert den momentan gesetzten Grafikmodus
GetImage	speichert den Inhalt eines rechteckigen Bildausschnitts bitweise in einer Puffervariablen
GetLineSettings	fragt die durch SetLineStyle gesetzten Parameter ab (Art, Muster und Dicke)
GetMaxMode	liefert die Nummer des *höchsten* Grafikmodus für den momentan aktiven Treiber
GetMaxX	liefert die größtmögliche X-Koordinate des momentan aktiven Grafikmodus zurück
GetModeName	liefert den Namen eines der Grafikmodi
GetPalette	ermittelt die momentan gesetzte Farb-Palette und ihre Größe
GetPaletteSize	liefert die Größe der momentan gesetzten Farb-Palette (d. h. Anzahl der Einträge/Farben)
GetPixel	liefert den Wert des Pixels auf den angegebenen Koordinaten zurück
GetTextSettings	fragt die durch SetTextStyle und SetTextJustify gesetzten Parameter ab (Zeichensatz, Stil und Justierung)
GetViewSettings	ermittelt die Grenzen des momentan gesetzten Grafikfensters und die Bedingungen für das Clipping
GetX	liefert die momentane X-Position des Grafik-Cursors
GetY	liefert die momentane Y-Position des Grafik-Cursors
GraphErrorMsg	liefert die zum angegebenen Fehlercode gehörige Meldung als String
GraphResult	liefert den momentanen Fehlerstatus des Grafikpakets als Intergerwert
ImageSize	liefert die Anzahl von Bytes zurück, die zur Speicherung eines bestimmten Ausschnitts mit GetImage gebraucht werden
InitGraph	initialisiert das Grafikpaket und schaltet in den Graphikmodus um
InstallUserDriver	zur Installation von eigenen Grafiktreibern
InstallUserFont	zur Installation von eignen Vektor-Zeichensätzen
Line	zeichnet eine Linie von (x1,y2) nach (x2,y2)
LineRel	zeichnet eine Linie relativ zur momentanen Position des Grafik-Cursors
LineTo	zeichnet eine Linie von der momentanen Position des Grafik-Cursors aus zur angegebenen Koordinate
MoveRel	bewegt den Grafik-Cursor von seiner momentanen Position aus um den angegebenen Betrag
MoveTo	setzt den Grafik-Cursor auf den angegebenen Punkt (x,y)
OutText	gibt einen String ab der momentanen Position des Grafik-Cursors aus
OutTextXY	gibt einen String ab der angegebenen Position aus
PieSlice	zeichnet einen Kreisausschnitt, verbindet seine Enden mit dem Mittelpunkt und füllt die entstandene Fläche *(Kuchenstück)*
PutImage	kopiert den Inhalt einer Puffervariablen in einen rechteckigen Ausschnitt des Bildschirms
PutPixel	zeichnet einen einzelnen Punkt auf den angegebenen Koordinaten
Rectangle	zeichnet ein Rechteck in der momentan gesetzten Farbe und der aktiven Linienart
RestoreCrt	setzt den Videomodus, der beim Start des Programms aktiv war
RestoreCrtMode	setzt den Videomodus, der beim Aufruf von InitGraph aktiv war
Sector	zeichnet ein ausgefülltes elliptisches *Kuchenstück*
SetActivePage	setzt eine Bildschirmseite für folgende Ausgaben
SetAllPalette	setzt sämtliche Einträge einer Farb-Palette neu
SetAspectRatio	setzt den Korrekturfaktor für das Höhen-/Seitenverhältnis des Bildschirms direkt
SetBkColor	setzt die Hintergrundfarbe für folgende Zeichenaktionen (und benutzt die aktive Farb-Palette dazu)
SetColor	setzt die Zeichenfarbe für folgende Zeichenaktionen (und benutzt die aktive Farbpalette dazu)
SetFillPattern	setzt ein benutzerdefiniertes Füll-Muster
SetFillStyle	setzt ein (vordefiniertes) Füll-Muster
SetGraphMode	setzt einen Grafikmodus und löscht den Bildschirm
SetLineStyle	setzt Breite und Muster für nachfolgende Linien
SetPalette	setzt einen Eintrag in einer Farb-Palette
SetRGBPalette	ändert einen Eintrag der momentan gesetzten Farb-Palette für den IBM-Adapter 8514 und für VGA-Karten
SetTextJustify	setzt Parameter für die Justierung von Textausgaben via OutText und OutTextXY
SetTextStyle	setzt einen Zeichensatz, den Stil und den Vergrößerungsfaktor
SetUserCharSize	legt voneinander unabhängige Vergrößerungsfaktoren in X- und Y-Richtung für Grafikzeichensätze fest
SetViewPort	setzt ein Zeichenfenster in der Grafik
SetVisualPage	bringt eine von mehrerern Grafikseiten auf dem Bildschirm zur Anzeige
SetWriteMode	legt fest, ob Linien-Zeichenoperationen den vorherigen Bildinhalt überschreiben oder als Verknüpfung ausgeführt werden
TextHeight	liefert die Höhe eines Strings (in Punkten)
TextWidth	liefert die Breite eines Strings (in Punkten)

GRAPH3

Routinen	
GraphColorMode	setzt den Modus *320∗200 Punkte, 4 Farben*
GraphMode	setzt den Modus *320∗200 Punkte, schwarzweiß*
HiRes	setzt den Modus *640∗200 Punkte, zweifarbig*
Palette	aktiviert die angegebene Farb-Palette
HiResColor	setzt eine Zeichenfarbe für den Modus HiRes
GraphBackground	setzt die Hintergrundfarbe für folgende Zeichenaktionen
GraphWindow	legt ein Zeichenfenster auf dem Bildschirm fest
Plot	zeichnet einen Punkt mit der angegebenen Farbe auf den angegebenen Koordinaten
Draw	zeichnet eine Linie zwischen zwei angegebenen Punkten in der angegebenen Farbe
ColorTable	definiert eine Farb-Übersetzungstabelle, über die die neue Farbe eines Punktes bei nachfolgenden Zeichenaktionen durch seine alte Farbe festgelegt wird
Arc	zeichnet einen Kreisausschnitt
Circle	zeichnet einen Kreis
GetPic	kopiert einen rechteckigen Ausschnitt des Bildschirms in eine Puffervariable
PutPic	kopiert den Inhalt einer Puffervariablen in den Bildschirm
GetDotColor	liefert die Farbe eines Punktes
FillScreen	füllt das gesamte aktive Fenster mit einer Farbe
FillShape	füllt eine umschlossene Fläche mit einer Farbe
FillPattern	füllt einen rechteckigen Bereich des Bildschirms mit dem momentan aktiven Muster in der angegebenen Farbe
Pattern	definiert ein Muster mit 8 × 8 Punkten, das von FillPattern verwendet wird

Turtlegrafik	
Back	bewegt den Zeichenstift um den angegebenen Betrag entgegen der momentanen Zeichenrichtung
Forwd	bewegt den Zeichenstift um den angegebenen Betrag in der momentanen Zeichenrichtung
ClearScreen	löscht den Inhalt des Zeichenfensters und setzt den Zeichenstift in die Mitte des Fensters
Heading	liefert die momentane Bewegungsrichtung des Zeichenstiftes
HideTurtle	läßt den Zeichenstift (die Schildkröte) vom Bildschirm verschwinden
Home	setzt den Zeichenstift auf seine Anfangsposition zurück
NoWrap	legt fest, daß der Zeichenstift nicht automatisch an der anderen Seite des Zeichenfensters wieder erscheint, wenn er eine Fensterkante überschritten hat
PenDown	setzt den Zeichenstift an, nachfolgende Bewegungen hinterlassen einen Strich
PenUp	hebt den Zeichenstift ab, nachfolgende Bewegungen hinterlassen keine Spuren
SetHeading	legt die Bewegungsrichtung fest
SetPencolor	legt die Zeichenfarbe fest
SetPosition	bewegt den Zeichenstift an die angegebene Position, ohne dabei zu zeichnen
ShowTurtle	läßt (den Zeichenstift) die Schildkröte auf dem Bildschirm erscheinen
TurnLeft	verändert die Bewegungsrichtung um den angegebenen Winkel gegen den Uhrzeigersinn
TurnRight	verändert die Bewegungsrichtung um den angegebenen Winkel im Uhrzeigersinn
TurtleWindow	legt ein Zeichenfenster auf dem Bildschirm fest
TurtleThere	prüft, ob der Zeichenstift sichtbar ist, und liefert ein boolsches Ergebnis
TurtleDelay	legt eine Verzögerung nach jedem Schritt des Zeichenstiftes fest
Wrap	legt fest, daß der Zeichenstift nach Überschreitung der Fenstergrenzen automatisch auf der gegenüberliegenden Seite wieder erscheint
XCor	liefert die momentane X-Koordinate des Zeichenstiftes
YCor	liefert die momentane Y-Koordinate des Zeichenstiftes

TURBO3

Routinen	
MemAvail	liefert die Gesamtgröße des freien Speicherbereichs auf dem Heap in Paragrafen (Einheiten zu je 16 Byte) als Integer zurück
MaxAvail	liefert die Größe des größten zusammenhängenden freien Speicherbereichs auf dem Heap in Paragrafen als Integer zurück
LongFileSize	liefert die Größe einer Datei im Format Real
LongFilePos	liefert die Position innerhalb einer Datei im Format Real
LongSeek	setzt die Position innerhalb einer Datei auf die angegebene Komponente und erwartet einen Realwert als Argument
HighVideo	setzt die Zeichenfarbe für Textausgaben bei Farbbildschirmen auf *gelb mit schwarzem Hintergrund*, bei monochromen Systemen auf *weiß mit schwarzem Hintergrund*
NormVideo	hat dieselbe Funktion wie HighVideo
LowVideo	setzt die Zeichenfarbe für Textausgaben auf *hellgrau mit schwarzem Hintergrund*

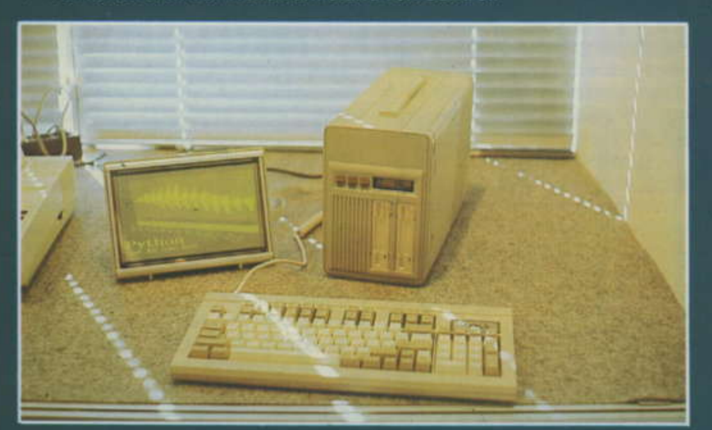
ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 11
S. 321–352



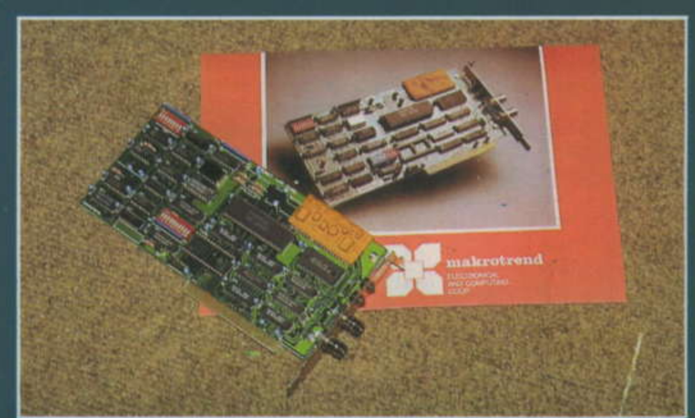
1 *Demonstration der Vielfalt am Stand von Controll*



2 *Laptop zur Programmierung von Werkzeugmaschinen*



3 *Transportabler PC MAT/H2 mit EL-Display*



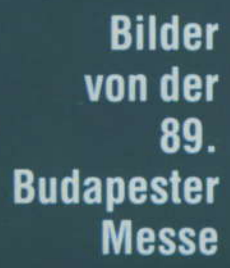
4 *ARCNET-kompatible LAN-Interfacekarte von Makrotrend*



5 *Unter dem Dach von Videoton: Vielzahl neuer Firmen*



6 *IBM Personal System/2 Model 50 Z*



Bilder von der 89. Budapester Messe

Lesen Sie dazu unseren Bericht in diesem Heft



7 *Future Technologie System AT-286/12*



8 *ASEM THOR 8070*



ZEITSCHRIFT FÜR MIKROELEKTRONIK · COMPUTERTECHNIK · INFORMATIK
MP
Heft 12 · 1989
Mikroprozessortechnik
VEB Verlag Technik Berlin
ISSN 0232-2892
ABT-2891
Echtzeitverarbeitung



1 **TEXTIMA-Beispiellösung für flexible Fertigung von Trikotagen**



2 **Datenbankgestütztes Entwicklungssystem PolyCAD**



3 ***Linotype Page Manager 410***



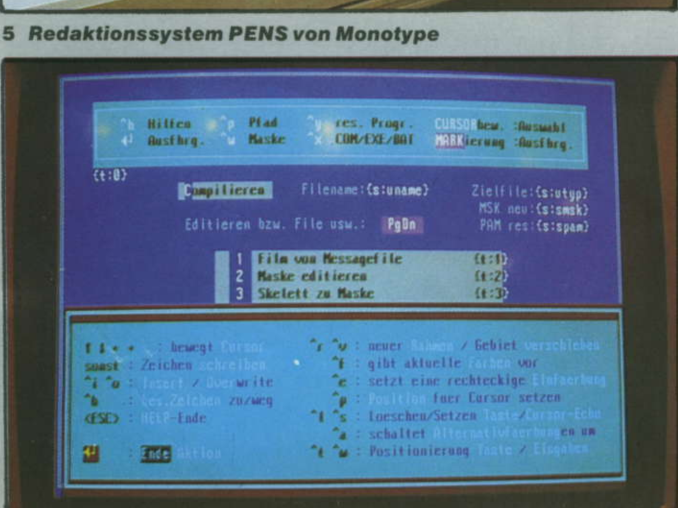
4 ***Linotype PostScript RIP 3***



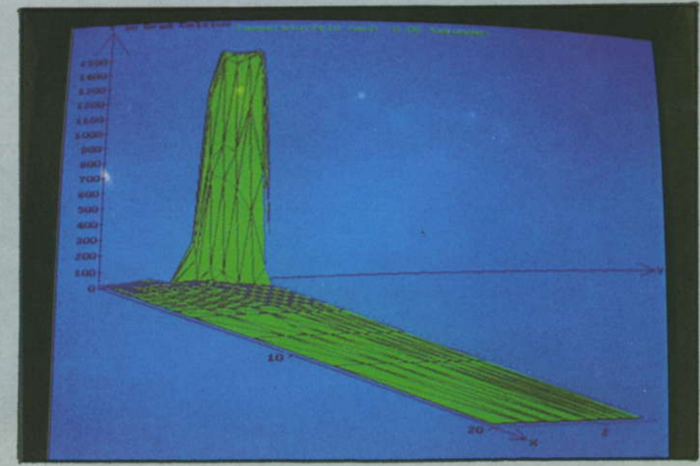
Lesen Sie hierzu unseren Bericht in diesem Heft.



5 ***Redaktionssystem PENS von Monotype***



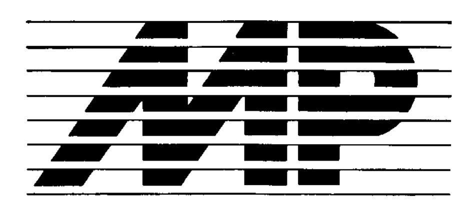
6 ***Ganzseitengestaltung mit PENS***



7 ***TINA programmiert die Dialoge mit dem PC***

8 ***ZTU-Software simuliert Schweißbarkeit und Schweißqualität***

Zeitschrift für Mikroelektronik · Computertechnik · Informatik
3. Jahrgang · ISSN 0232–2892



Mikroprozessortechnik, Heft 12 · 89

Herausgeber Kammer der Technik, Fachverband Elektrotechnik

Verlag VEB Verlag Technik, Oranienburger Str. 13/14, DDR – 1020 Berlin; Telegrammadresse: Technikverlag Berlin; Telefon: 28700, Telex: 0112228 techn dd

Verlagsdirektor Klaus Hieronimus

Redaktion Hans Weiß, Verantwortlicher Redakteur (Tel. 2870371); Redakteure: Herbert Hemke (Tel. 2870203), Hans-Joachim Hill (Tel. 2870209); Sekretariat Tel. 2870381

Gestaltung Christina Bauer

Titel Frank-Norbert Beyer

Lizenz-Nr. 1710 des Presseamtes beim Vorsitzenden des Ministerrates der Deutschen Demokratischen Republik

Gesamtherstellung Druckerei Märkische Volksstimme Potsdam

Redaktionsschluß 16. Oktober 1989

AN (EDV) 49837

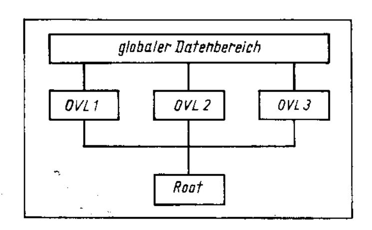
Erscheinungsweise monatlich 1 Heft

Heftpreis 5,– M, Abonnementspreis vierteljährlich 15,– M; Auslandspreise sind den Zeitschriftenkatalogen des Außenhandelsbetriebes BUCHEXPORT zu entnehmen.

Bezugsmöglichkeiten
DDR: sämtliche Postämter; *SVR Albanien:* Direktorije Quendrore e Perhapjes dhe Propaganditit te Librit Rruga Konference e Pezes, Tirana; *VR Bulgarien:* Direkzia R.E.P., 11a, Rue Paris, Sofia; *VR China:* China National Publications Import and Export Corporation, West Europe Department, P.O. Box 88, Beijing; *ČSSR:* PNS – Ustřední Expedicia a Dovoz Tísku Praha, Slezská 11, 120 00 Praha 2, PNS, Ústredna Expedicia a Dovoz Tlače, Pošta 022, 885 47 Bratislava; *SFR Jugoslawien:* Jugoslovenska Knjiga, Terazija 27, Beograd; Izdavačko Knjižarsko Produzeće MLADOST, Ilica 30, Zagreb; *Koreanische DVR:* CHULPANMUL Korea Publications Export & Import Corporation, Pyongyang; *Republik Kuba:* Empresa de Comercio Exterior de Publicaciones, O'Reilly No. 407, Ciudad Habana; *VR Polen:* C.K.P.i.W. Ruch, Towarowa 28, 00-958 Warszawa; *SR Rumänien:* D.E.P. Bucureşti, Piaţa Scînteii, Bucureşti; *UdSSR:* Sämtliche Abteilungen von Sojuzpečat' oder Postämter und Postkontore; *Republik Ungarn:* P.K.H.I., Külföldi Előfizetési Osztály, P.O. Box 16, 1426 Budapest; *SR Vietnam:* XUNHASABA, 32, Hai Ba Trung, Há Nôi; *BRD und Berlin (West):* ESKABE Kommissions-Grossobuchhandlung, Postfach 36, 8222 Ruhpolding/Obb.; Helios-Literatur-Vertriebs-GmbH, Eichborndamm 141–167, Berlin (West) 52; Kunst und Wissen Erich Bieber OHG, Postfach 46, 7000 Stuttgart 1; Gebrüder Petermann, BUCH + ZEITUNG INTERNATIONAL, Kurfürstenstraße 111, Berlin (West) 30; *Österreich:* Helios-Literatur-Vertriebs-GmbH & Co. KG, Industriestraße B 13, 2345 Brunn am Gebirge; *Schweiz:* Verlagsauslieferung Wissenschaft der Freihofer AG, Weinbergstr. 109, 8033 Zürich; *Alle anderen Länder:* örtlicher Fachbuchhandel; BUCHEXPORT Volkseigener Außenhandelsbetrieb der Deutschen Demokratischen Republik, Postfach 160, DDR – 7010 Leipzig und Leipzig Book Service, Talstraße 29, DDR – 7010 Leipzig



Der Beitrag auf der **Seite 355** befaßt sich mit Echtzeitbetriebssystemen für 16-Bit-Rechner. Sie schaffen, gemeinsam mit der entsprechenden Hardware, die Voraussetzungen für die schnelle Reaktion des Rechners auf äußere Einflüsse – im allgemeinen im Millisekundenbereich liegend. Auf unser Titelbild bezogen, werden Sie uns vielleicht recht geben, daß wir den Vergleich mit dem Fahren auf unseren Straßen nicht zu weit hergeholt haben.



Erfahrungen bei der Programmierung komplexer Programmsysteme in C vermittelt der Beitrag „Programmsysteme in C auf 8-Bit-Rechnern" auf der **Seite 363**. Dabei wird unter anderem ein Programmsystem vorgestellt, das sich in vier komplexe Teilprogramme gliedert – die Wurzel des Systems (Root) mit dem Rahmenprogramm und die drei Teilprogramme mit aufgabenspezifischen Überlagerungsstrukturen.

Vorschau

Für MP 1/1990 bereiten wir für Sie Beiträge zu folgenden Themen vor:
- Zuverlässigkeit mit Einchipmikrorechnern
- Eingabemasken mit Turbo-Pascal
- Farbgrafikadapter für K 1520-Systeme

Inhalt

1 000. Bildungscomputer übergeben

Unmittelbar vor Beginn des neuen Schuljahres lösten die Werktätigen des VEB Robotron-Meßelektronik „Otto Schön" Dresden und ihre Kooperationspartner in Sömmerda und Berlin ihr Versprechen gegenüber dem Bildungswesen ein. Im Computerkabinett der Erweiterten Oberschule „Albert Einstein" Berlin-Marzahn übergab der Dresdner Betriebsdirektor Dr. Hans Fischer am 30. August sieben Bildungscomputer A 5105, darunter den 1 000. Bis zum Jahresende sollen es 5 000 sein, die an das Bildungswesen geliefert werden.
„Ich denke, daß wir dem Direktor der Schule heute Technik zur Verfügung stellen, die alle Ansprüche an eine solide Informatikausbildung erfüllt. Wir hoffen, viele junge Leute werden an der Informatik und an der Rechentechnik Spaß finden", sagte Dr. Fischer. Letztlich solle auch durch das Beschäftigen mit der Informatik Engagement und Kreativität der Schüler gefördert werden.
Geplant ist, alle erweiterten Oberschulen mit jeweils sieben Bildungscomputern auszustatten. *ADN*

Diskettenlaufwerke für den Bildungscomputer

Mehr als 1 000 Diskettenspeichereinheiten für den Bildungscomputer von Robotron haben die Kollektive des VEB Elektroprojekt und Anlagenbau Berlin (EAB) bisher vertragsgerecht produziert. Das berichtete die Berliner Zeitung anläßlich der 14. Berliner Bestarbeiterkonferenz in ihrer Ausgabe vom 14. September. Weiter heißt es: Auf dem Wege steigender Monatsleistungen wollen die Beschäftigten aus dem Bereich Konsumgüterfertigung des EAB bis zum Jahresende 5 000 dieser Computerkomponenten ausliefern. Damit steigern sie die Produktion von Erzeugnissen für den Bevölkerungsbedarf auf insgesamt 66 Millionen Mark – fast ein Drittel mehr als im vergangenen Jahr.
Die Diskettenspeichereinheit, Ende Juni in die Serie übergeleitet, trägt seit August das Gütezeichen „Q". „Wir erweitern damit unser Konsumgüterprogramm um ein Erzeugnis, das unserer Verantwortung als Hersteller von mikroelektronischer Automatisierungstechnik entspricht. Um dem großen Bedarf des Bildungswesens an Geräten für die Informatikausbildung zu entsprechen, werden wir unsere Leistungen weiter steigern. So sind die Montagekollektive vor einiger Zeit zur Arbeit in zwei Schichten übergegangen", berichtete Dr. Roland Wuttke, Direktor des Bereiches Konsumgüter des EAB. *MP*



RFT-Service in Moskau

Am 19. September 1989 bestand das „Technische Zentrum des Kundendienstes für Erzeugnisse der Nachrichtentechnik der DDR in Moskau" fünf Jahre. Das Technische Zentrum kann sich auf gute Erfahrungen des „Büros für die technische Wartung der Fernmeldetechnik" berufen, das von 1981 bis 1984 bestand.
Der Kundendienst für Erzeugnisse der RFT-Nachrichtentechnik begann bereits 1963 mit der Bildung des Technisch-Kommerziellen Büros der damaligen VVB Nachrichten- und Meßtechnik.
Das Jubiläum des Technischen Zentrums fällt zeitlich mit der 26jährigen Wiederkehr der Einrichtung des Technisch-Kommerziellen Büros zusammen.
Mit Bildung des Technischen Zentrums – 1984 – wurde ein neues effektives System der Zusammenarbeit wirksam. Damit wurde eine Stätte der Begegnung, des Erfahrungsaustausches und der Zusammenarbeit direkt beim bedeutendsten Handelspartner des Kombinates Nachrichtenelektronik geschaffen, um gemeinsam das bis ins Jahr 2 000 reichende Programm der Zusammenarbeit in Wissenschaft, Technik und Produktion mit vielfältigen Ideen und Beispielen zu erfüllen.
Die Aufgaben und Leistungen des Technischen Zentrums beinhalten die Beratung des Kunden, die Untersuchung der Fernmeldenetze, die Ausarbeitung von Projekten, die Lieferung von Nachrichtenmitteln, ihre Montage und Inbetriebnahme sowie die Wartung der Ausrüstungen und die Schulung des Personals der Kunden. Es wurden zahlreiche Schulungen für Anwender, Symposien und Erfahrungsaustausche mit Spezialisten, Approbationen und Pressekonferenzen organisiert, auf denen neue Erzeugnisse und Technologien vorgestellt, erläutert und Problemdiskussionen zwischen Wissenschaftlern und Technikern geführt wurden. Eine ständige Ausstellung in den Räumen des Technischen Zentrums gibt den Besuchern einen aktuellen Überblick über das Produktionsprofil, Neu- und Weiterentwicklungen, Forschungsergebnisse und Ergebnisse der zweiseitigen und mehrseitigen Zusammenarbeit im Rahmen des RGW.
Das Kombinat hat enge Verbindungen zum aufnahmefähigen und ständig wachsenden Absatzmarkt der UdSSR, in den es jährlich ein Drittel seiner Gesamtproduktion exportiert. Umfangreiche Beziehungen bestehen mit Betrieben, mit denen direkt Produktionsbeziehungen angebahnt wurden. *MP*

Produktion von Computertastaturen

Der VEB Robotron-Elektroschaltgeräte Auerbach stellt sich auf die Produktion von Tastaturen für die Computer ein, die mit den künftigen 32-Bit-Mikroprozessoren ausgerüstet werden. Die Leistungen der Forscher des Kombinates Mikroelektronik bei der kurzfristigen Entwicklung des neuen 32-Bit-Mikroprozessor-Systems beflügeln die fast 1 500 Werktätigen dieses Zulieferbetriebes des Kombinates Robotron, gleichfalls wichtige Entwicklungsarbeiten den volkswirtschaftlichen Erfordernissen entsprechend abzuschließen, damit planmäßig eine neue Rechnergeneration auf den Markt gebracht werden kann. Darüber informierten Betriebskollektive im August den Minister für Elektrotechnik und Elektronik, Felix Meier, der mit Arbeitern und Ingenieuren über die Erfüllung der diesjährigen Aufgaben und über die Vorbereitung des Planjahres 1990 beriet. Die Fertigung von Computertastaturen steige nach Stückzahlen gegenüber 1989 um 41 Prozent. Auch die anderen Fertigungsbereiche, in denen Taster, Mikroschalter und elektrotechnische Steuerungen hergestellt werden, hätten sich auf einen hohen Leistungsanstieg eingestellt. *ADN*

Delegationen der Planungsorgane der DDR und UdSSR in Kombinaten

Die Delegationen der Zentralen Planungsorgane der DDR und der UdSSR machten sich im Juli in den Dresdner Kombinaten Robotron und GERMED mit Ergebnissen und Vorhaben der wirtschaftlichen und wissenschaftlich-technischen Zusammenarbeit zwischen beiden Ländern vertraut. Bei Robotron besichtigten die vom Kandidaten des Politbüros des ZK der SED, Gerhard Schürer, Stellvertreter des Vorsitzenden des Ministerrates und Vorsitzender der Staatlichen Plankommission der DDR, und vom Kandidaten des Politbüros des ZK der KPdSU, Juri Masljukow, Erster Stellvertreter des Vorsitzenden des Ministerrates und Vorsitzender des Staatlichen Plankomitees der UdSSR, geleiteten Delegationen Produktionsstätten des Kombinats-Stammbetriebes. So für die Herstellung der Arbeitsplatzcomputer sowie ein Prüffeld für elektronische Datenverarbeitungsanlagen und 32-Bit-Rechentechnik. Robotron arbeitet seit langem eng mit sowjetischen Partnern auch auf dem Gebiet der Schlüsseltechnologien zusammen, zu denen die Direktbeziehungen zwischen Robotron-Projekt Dresden und Zentrprogrammsystem Kalinin bei der gemeinsamen Entwicklung von Datenbanksystemen gehört.
Bei ihrem Rundgang erfuhren die Gäste, daß das Kombinat Robotron einen beträchtlichen Teil seines Exports in die Sowjetunion liefere, wozu in diesem Jahr 24 000 Personalcomputer, 50 EDV-Anlagen und speziell für die UdSSR entwickelte und produzierte Steuerrechnersysteme NEWA für Telefonzentralen zählen.
ADN

Liebe Leserinnen und Leser,
unser Redaktionssekretariat ist zur Zeit nicht besetzt. Wir bitten Sie daher um Verständnis, wenn die Kommunikation zwischen Ihnen und uns nicht immer wie gewohnt möglich ist. Vielleicht können aber gerade Sie uns unterstützen, indem Sie sich bei uns als redaktionelle Mitarbeiterin bzw. als redaktioneller Mitarbeiter bewerben oder Bekannte über unser Stellenangebot informieren.
Sollten Sie also im Raum Berlin wohnen und an einer vielseitigen Tätigkeit in einem kleinen Kollektiv interessiert sein, erwarten wir ihre Anfragen oder Bewerbungen telefonisch – unter 2 87 03 71 – bzw. schriftlich – VEB Verlag Technik, Redaktion MP, Oranienburger Straße 13/14, Berlin, 1020.
Ihre Redaktion MP

16-Bit-Betriebssysteme der Echtzeitverarbeitung

Wolf-Dieter Hardt, Erfurt

Anforderungsdefinition

Die Einführung von Computer Integrated Manufacturing (CIM) erfordert die verstärkte Integration von Rechentechnik in fertigungstechnische Basissysteme. Diese Rechentechnik erhält über Sensoren Eingangsinformationen, die auf der Grundlage applikationsspezifischer Überführungs- oder Ergebnisfunktionen /1/ in Ausgabeinformationen transformiert und über Aktoren dem fertigungstechnischen Basissystem zur Verfügung gestellt werden. Die Arbeitsweise dieser Rechentechnik wird als prozeßgekoppelter Betrieb bezeichnet. Die rechnerinterne Darstellung der applikationsspezifischen Überführungs- oder Ergebnisfunktionen erfolgt durch Rechenprozesse. Unter einem Rechenprozeß versteht man die auf seine Lebensdauer beschränkte Abwicklung eines Programms auf einem Rechnersystem. Die Lebensdauer eines Rechenprozesses beginnt mit der Anmeldung und endet mit seiner Abmeldung. Damit existieren Rechenprozesse nicht nur während ihrer tatsächlichen Bearbeitung durch die CPU des Rechnersystems, sondern schon vor Beginn der Bearbeitung und während eingeplanter oder erzwungener Wartezeiten. In der praktischen Anwendung von prozeßgekoppelten Rechnersystemen ist im allgemeinen die Anzahl der zu verwaltenden Rechenprozesse um ein Vielfaches größer als die Anzahl der CPUs des Rechnersystems. Daraus resultierend müssen parallele Rechenprozesse verwaltet werden, die um die Nutzung der CPU konkurrieren. Neben dieser Konkurrenz ist zu erwarten, daß die Rechenprozesse zur Erfüllung ihrer Aufgaben kooperieren. Die Prinzipien dieser Kooperation lassen sich in die Aufgabenklassen Ereignissynchronisation, gemeinsame Betriebsmittelbenutzung und Kommunikation einteilen. Wird durch ein Betriebssystem sichergestellt, daß die Aufgaben der Konkurrenz und Kooperation paralleler Rechenprozesse zeitsynchron zu den Bedingungen des fertigungstechnischen Basissystems erfüllt werden, arbeitet das Betriebssystem im Echtzeitbetrieb. Im allgemeinen werden solche Betriebssysteme als Echtzeitbetriebssysteme bezeichnet, die auf asynchrone Anforderungen des Basissystems innerhalb weniger Millisekunden die zugeordneten Rechenprozesse zur Abarbeitung anmelden.



Echtzeituhr
Zeichnung: Steger

Eine Analyse der internationalen Entwicklungstendenzen bei Betriebssystemen der Echtzeitverarbeitung zeigt nachfolgende vier Haupttrends, deren Grenzen sich überlappen und deren inhaltliche Realisierungsniveaus vom einfachen Betriebssystemkern bis zu komfortabel ausgebauten Betriebssystemen für die Programmentwicklung und -ablaufsteuerung reichen.

① Entwicklung modularer, generierbarer und effizienter Betriebssystemkerne, die applikationsspezifisch durch wahlweise Zuordnung weiterer Systemschichten (E/A-Organisation, Mensch-Maschine-Interface, Entwicklungswerkzeuge) ein möglichst universelles Einsatzspektrum abdecken.
Hauptvertreter dieser Systeme sind unter anderem die Betriebssystemfamilien iRMX /2/, /3/ und RMOS /4/.

② Die Nutzung von Hochsprachen wie Basic, Pascal, C, Forth und beginnend Modula-2 /5/ für die Echtzeitverarbeitung führte sowohl zur Bereitstellung von portierbaren Betriebssystemkernen in Hochsprachen als auch zur Erweiterung der Sprachentwicklungsumgebungen durch Bereitstellung von diesem Umfeld angepaßten echtzeitfähigen Betriebssystemkernen (Micropower/Pascal, multiFORTH /6/, RTOS-UH/Pearl).

③ Die Verlagerung der Funktionen der Echtzeitdatenverarbeitung ins Silizium beinhaltet die Qualitätsmerkmale, ausgehend von PROM-residenten Betriebssystemen (VRTX) und Koprozessoren (i80130) bis hin zu speziellen Mikroprozessoren (Transputersystemfamilie), und führt konsequent weiterentwickelt zu einer dem Aufgabenprofil angepaßten Mikroprozessorarchitektur.

④ Entwicklung und Abarbeitung von Echtzeitapplikationen im Umfeld international marktführender Betriebssysteme, wie CP/M, MS-DOS, UNIX oder VMS. Hierbei werden nicht nur Betriebssysteme hinsichtlich der Echtzeitfähigkeit erweitert (HP-UX, Concurrent CP/M-86), sondern auch speziell auf die Entwicklungsumgebung abgestimmte Echtzeitbetriebssysteme bereitgestellt (MTOS-UX, V60 RTOS, VAXELN).

Nationaler Stand

Entsprechend der internationalen Breite des Angebotes an Betriebssystemen der Echtzeitverarbeitung ist auch im nationalen Vergleich eine gewisse Vielfältigkeit festzustellen. Der Hauptanteil hinsichtlich praxiswirksamer Einsatzfälle der 16-Bit-Leistungsklasse wird durch die Betriebssysteme EBS /7/, IRTS /8/, /9/, EMOS /10/ und BOS 1810 /11/ abgedeckt.
Das System EBS der 16-Bit-Leistungsklasse

Wolf-Dieter Hardt (39) studierte von 1970 bis 1974 an der Karl-Marx-Universität Leipzig in der Fachrichtung Mathematik. Seit 1974 arbeitet er im Kombinat Robotron und beschäftigt sich mit der Entwicklung und dem Einsatz von Echtzeitbetriebssystemen, die auch einen Schwerpunkt seiner gegenwärtig laufenden außerplanmäßigen Aspirantur an der Technischen Hochschule Ilmenau bilden.

ist ein Echtzeitkern, der in der Zielsetzung der Gruppe ③ zuzuordnen ist. Die Generierung des EBS (Komponentenauswahl, Speicherplatzzuordnung) erfolgt beim Entwickler. EBS belegt zirka 5,5 KByte, ist für den U 8001 geeignet und beinhaltet keine E/A-Geräteunterstützung im Sinne der Bereitstellung vorgefertigter Gerätetreiber. Eine nachträgliche Nutzeranpassung ist über einen speziellen Parameterbereich möglich.
Die Systeme IRTS, EMOS und BOS 1810 sind in die Gruppe ① einzustufen, unterscheiden sich aber erheblich im Umfang der bereitgestellten Erweiterungsschichten.
IRTS ist ein generierbarer Echtzeitkern (zirka 4 KByte) für den Prozessor U 8001/2 mit den Erweiterungsschichten Basis-E/A-System (Terminal, Drucker, Monitor) sowie Debugger. Der Nutzer von IRTS kann mittels des Generierungsprogramms ICL einsatzspezifische Betriebssystemversionen erzeugen, die vorrangig für OEM-Einsätze in eingelagerten Mikrorechnern vorgesehen sind.
EMOS wurde für den OEM-Echtzeitteil des Industriecomputers ICA 710 bereitgestellt. Darüber hinaus enthält der ICA einen PC-Teil, der unter Steuerung von MS-DOS arbeitet. Beide Rechnerbaugruppen sind durch zwei serielle Schnittstellen lose gekoppelt. Das Echtzeitbetriebssystem EMOS besitzt einen Echtzeitkern mit angelagerten Funktionsmodulen (Debugger, Systemreporter, Fehleranzeige), E/A-Gerätetreibern (Prozeß-E/A, V24- und IFSS-Interface) und einem File-Subsystem für den Zugriff auf die Massenspeicher des PC-Teils unter EMOS.
BOS 1810 ist ein komfortables, inhaltlich voll ausgebautes Echtzeitbetriebssystem, das sowohl als Softwareentwicklungssystem (gegenwärtig A 7100/7150) als auch für OEM-Einsätze auf der Basis des Prozessors K 1810 WM86 genutzt werden kann. Der Speicherbedarf des Kerns liegt im Minimum bei zirka 19 KByte und kann durch wahlweise Nutzung umfangreicher Ergänzungsschichten (E/A-Systeme, Anwendungslader, Human Interface, Universelles Entwicklungsinterface) weiter ausgebaut werden. Letztere Ergänzungsschicht eröffnet Möglichkeiten zur Nutzung umfangreicher Softwareentwicklungswerkzeuge auf Assembler- und Hochsprachniveau (Fortran 77, PL/M 86) sowie zur maschinellen Generierung applikationsspezifischer BOS 1810-Versionen.

Eigenschaften ausgewählter Betriebssysteme

Systemarchitektur

Ausgehend von den Arbeiten von Giloi /12/ und Horn /13/, wird die Systemarchitektur eines Echtzeitbetriebssystems aufgefaßt als das Tupel
Systemarchitektur = (Operationsprinzip, Softwarestruktur).
Das Operationsprinzip definiert das funktionelle Verhalten des Betriebssystems und die Softwarestruktur die Art und die Anzahl der

funktionellen Schichten und ihrer Relationen. Nachfolgende Ausführungen beschränken sich auf vergleichende Untersuchungen des funktionellen Verhaltens der Kerne der Echtzeitbetriebssysteme EBS, IRTS, EMOS und BOS 1810. Als Kern eines Echtzeitbetriebssystems werden alle die Rechenprogramme zusammengefaßt, die entsprechend den Anforderungen der Konkurrenz und Kooperation die Verwaltung paralleler Rechenprozesse ermöglichen. Das Operationsprinzip des Kerns eines Echtzeitbetriebssystems wird definiert als das Tupel

Operationsprinzip = (Funktionsprinzip, Objekt).

Wesentliche Funktionsprinzipien der Echtzeitkerne sind

* Verwaltung von Rechenprozessen
* Ereignissynchronisation
* Verwaltung von Betriebsmitteln
* Nachrichtenkommunikation.

Im Kontext der Echtzeitbetriebssysteme wird unter einem Objekt die zielgerichtete Zusammenfassung von prozessorspezifischen Datentypen (Byte, Word, Pointer, ...) aufgefaßt. Der konkrete Aufbau der objektspezifischen Datenstruktur kann für den Nutzer abstrakt bleiben. Die Arbeit mit den Objekten erfolgt mittels über ihnen definierter Operationen, die den Anforderungen der Konkurrenz und Kooperation paralleler Rechenprozesse Rechnung tragen. Diese Operationen werden als Systemdienste bezeichnet.

Wesentliche Gütekriterien eines Echtzeitbetriebssystems sind die Art der verfügbaren Objekte, die Verfahren ihrer Definition und des Löschens (dynamisch = im laufenden Betrieb, statisch = bei Generierung) sowie die Funktionalität und die Anzahl der verfügbaren Systemdienste. Eine Übersicht über benutzte Objekttypen (für EBS und BOS 1810 als Auswahl) zeigt Tafel 1.

Tafel 1 Objekte der Echtzeitkerne EBS, IRTS, EMOS und BOS 1810

Betriebssystem	Task	Event	Semaphore	Region	Segment	Message	Mailbox
EBS	x (10)		x (4)	x (5)	x (2)		x (5)
IRTS	x (12)		x (6)		x (3)	x (7)	x (7)
EMOS	x (8)	x (5)	x (2)	x (1)	x (2)		x (2)
BOS 1810	x (7)		x (4)	x (5)	x (3)		x (4)

( ) Anzahl der Systemdienste über dem Objekttyp

Die Verwaltung von Rechenprozessen

Die hier in der Analyse einbezogenen Echtzeitbetriebssysteme verwalten Rechenprozesse in Form von
Task-Objekten
Interrupt-Handlern
Exception-Handlern.
Entsprechend dieser Einteilung unterscheiden sich die Methoden ihrer Verwaltung.

Task-Objekt

Das Task-Objekt ist aufzufassen als eine Verwaltungseinheit eines Rechenprozesses und der für seine Beschreibung erforderlichen Attribute. Diese Attribute werden als Tasksteuervektor bezeichnet. Die Definition eines Rechenprozesses impliziert, daß ein Taks-Objekt verschiedene Systemzustände annehmen kann, die insbesondere durch ihre Beziehung zu globalen Betriebsmitteln des Rechnersystems bestimmt sind. Im einfachsten Fall verwaltet das Betriebssystem drei Taskzustände durch eine Verkettung der zugeordneten Tasksteuervektoren.

bereit Die Task besitzt alle erforderlichen Betriebsmittel, wartet jedoch noch auf die Zuteilung des Betriebsmittels CPU.

aktiv Die Task besitzt die CPU, und ihr Rechenprozeß kann forschreiten.

gestoppt Die Task wurde infolge gegenwärtig nicht erfüllbarer Bedingungen von der Weiterarbeit ausgeschlossen.

Tafel 2 zeigt die Anzahl der realen Taskzustände der vorgestellten Betriebssysteme. Die Vergrößerung der dort implementierten Zustände ergibt sich im wesentlichen aus einer weiteren Unterteilung des Zustandes *gestoppt*.

Tafel 2 Taskzustände der Echtzeitkerne EBS, EMOS und BOS 1810

Betr[illegible]	[illegible] der Zustände	[illegible]	[illegible]	[illegible]estoppt		[illegible]
[illegible]	[illegible]	x	x	x	x	x
EMOS	4	x	x	x	x	
BOS 1810	5	x	x	x	x	x

Die Anzahl der Tasks, welche gleichzeitig *aktiv* sein können, wird durch die Anzahl der durch das Betriebssystem verwalteten CPUs bestimmt. Auswahl und Zuordnung einer Task zu einer CPU übernimmt das Kernprogramm Scheduler auf der Grundlage der sich im Zustand *bereit* befindlichen Tasks.

Für die Beurteilung der Echtzeitfähigkeit eines Kerns sind neben dem Auswahlalgorithmus insbesondere die Gründe für die Aktivierung des Schedulers von wesentlicher Bedeutung.

Die hier betrachteten Echtzeitkerne realisieren in erster Linie übereinstimmend einen dringlichkeitsgesteuerten Auswahlalgorithmus, in dem jeder Task eine Priorität zugeordnet wird. Erfolgt die Zuordnung zum Zeitpunkt der Definition der Task und ist nicht veränderbar, spricht man von statischer Priorität (EBS), lassen sich festgelegte Prioritäten verändern (IRTS, EMOS, BOS 1810), von dynamischer Priorität.

Der Rechenprozeß einer Task, der die CPU besitzt, arbeitet so lange, bis er entweder einen in Anspruch genommenen Systemdienst beendet hat oder im Ergebnis eines Interrupts eine Task höherer Priorität in den Zustand *bereit* eingeordnet wurde. In beiden Fällen entscheidet der Scheduler, ob die *aktive* Task entsprechend ihrer Priorität und den Ergebnissen des benutzten Systemdienstes die CPU behalten kann oder ob eine ihrer Prioritäten nach wichtigere Task ihre Abarbeitung weiterführen muß. Im letzteren Fall muß die *aktive* Task die CPU räumen, das heißt, sie wird verdrängt, um der höherpriorisierten Task die CPU zu überlassen. In Abhängigkeit von der konkreten Verdrängungsursache wird der entsprechende Steuervektor einem Taskzustand zugeordnet. Diese Schedulerstrategie bezeichnet man als aperiodisch verdrängend.

Da alle hier betrachteten Echtzeitkerne gleiche Prioritäten für verschiedene Tasks zulassen, besitzen deren Scheduler für den Fall, daß derartige Tasks gleichzeitig im Zustand *bereit* sind, einen weiteren Auwahlalgorithmus.

So arbeitet beispielsweise IRTS weiterhin verdrängend, jedoch wird bereits bei der Generierung der Task festgelegt, für welches Zeitintervall diese Task im Konfliktfall die CPU erhalten soll. Diese Strategie wird als periodisch verdrängend bezeichnet.

EBS und BOS 1810 lösen diesen Konflikt über eine FIFO-Warteschlange, indem gleiche Prioritäten in der Reihenfolge ihres Eintrages der CPU zugeordnet werden. Allerdings können die in der Warteschlange noch vorhandenen Tasks nur dann die CPU erhalten, wenn sich die *aktive* Task selber verdrängt oder durch eine höherpriorisierte Task die CPU entzogen und erneut in die FIFO-Warteschlange eingeschrieben wird.

Scheduleralgorithmen, in denen die Task ausschließlich selber entscheidet, unter welchen Bedingungen sie die CPU freigibt, werden als nicht verdrängend bezeichnet. Diese Algorithmen sind im allgemeinen für den Echtzeitbetrieb ungeeignet und finden in modernen Kernen nur noch bei der Lösung des dargestellten Konfliktes Anwendung.

Die benötigte Zeitdauer einer Taskumschaltung, das sogenannte context switching, das als ein entscheidendes Gütemerkmal eines Echtzeitkerns angesehen wird, kann bei Realisierung durch Software exakt nur für klar definierte Systemzustände angegeben werden. Allgemein kann eingeschätzt werden, daß die Zeitdauer von der CPU, ihrer Taktfrequenz und der internen Organisation der durch den Kern verwalteten Taskwarteschlangen beeinflußt wird. Die besonderen Vorteile spezieller Echtzeitprozessoren der in der Einleitung dargestellten Gruppe ③ liegen neben anderen auch in einer Minimierung der Zeitdauer der Taskumschaltung. Hier werden Zeiten von 1 μs (SAB 80199 /14/) bis zu < 1 μs (T412 /15/) erreicht.

Interrupt-Handler

Interrupt-Handler gestatten einem Echtzeitkern die Reaktion auf Ereignisse des prozeßgekoppelten Systems. Die Zuordnung dieser Rechenprozesse zur CPU erfolgt durch Mechanismen der Hardware asynchron zur Abarbeitung der durch Task-Objekte verwalteten Rechenprozesse. Daraus resultierend müssen Interrupt-Handler vor Beginn und vor Abschluß ihrer Aktionen den Registerzustand der CPU eigenverantwortlich retten und regenerieren. Die Benutzung von Systemdiensten des Kerns ist nur eingeschränkt möglich.

Ein Rechenprozeß wird durch die Eintragung seiner Startadresse in die Interruptvektortabelle der CPU zum Interrupt-Handler. Diese Aktion und das Löschen der Startadresse in der Vektortabelle übernehmen im EBS und im BOS 1810 Systemdienste des Kerns, die von durch Task-Objekte verwalteten Rechenprozessen benutzt werden können. Im IRTS und im EMOS erhält der Anwender keine Kernunterstützung, so daß hier eine statische Zuordnung von Interrupt-Handlern in der Phase der Programmentwicklung vorgenommen werden muß.

Ein weiterer Aspekt der Arbeit mit Interrupt-Handlern ist die Möglichkeit ihrer Synchronisation mit durch das Task-Objekt verwalteten Rechenprozessen. Diese Synchronisation kann direkt unter Benutzung des Task-Objektes – wie im BOS 1810 – oder unter Zwi-

schenschaltung von Synchronisationsobjekten, wie dem Semaphore im EBS, erfolgen.

Exception-Handler
Die Aufgabe eines Exception-Handlers besteht in der Bearbeitung der Fehler, die Rechenprozesse der Task-Objekte bei der Benutzung von Systemdiensten erzeugen. Exception-Handler werden nur im BOS 1810 unterstützt. Ihre Aktivierung erfolgt durch den Echtzeitkern als Prozedur des den Fehler auslösenden Rechenprozesses einer Task. Im BOS 1810 kann per Definition festgelegt werden, unter welchen Bedingungen und an welchen Exception-Handler die Steuerung im Fehlerfall übergeben wird.

Ereignissynchronisation

Dieses Funktionsprinzip umfaßt alle die Mechanismen des Echtzeitkerns, die eine Synchronisation der Abarbeitung von Rechenprozessen in Abhängigkeit von Ereignissen ermöglichen. Die bei der Synchronisation beteiligten Rechenprozesse werden eingeteilt in solche, die mittels des Systemdienstes *signalisieren* ein Ereignis produzieren und in solche, die mittels des Dienstes *erwarten* die Konsumtion eines Ereignisses beantragen. Der Echtzeitkern EMOS stellt den Objekttyp Event zur Ereignissynchronisation zur Verfügung. In den Kernen EBS, IRTS und BOS 1810 muß das Semaphore-Objekt benutzt werden.
Der Systemdienst *signalisieren* des EMOS führt zu einer Registrierung des Ereignisses in der Datenstruktur des Event-Objektes, wenn noch kein Konsument dieses erwartet. Nach der Registrierung kann der Produzent seine Arbeit weiterführen. Wartet bereits eine Task auf dieses Ereignis, wird dieser Konsument aus der Warteschlange des Event befreit und gemeinsam mit der Produzent-Task in den Zustand *bereit* eingeordnet, und es wird zum Scheduler verzweigt.
Der Systemdienst *erwarten* des EMOS kettet für den Fall den Konsument in die Warteschlange des Event ein, daß kein Ereignis in der Datenstruktur des Event registriert wurde. Damit wird der Konsument in den Zustand *gestoppt* eingeordnet und von der Weiterarbeit ausgeschlossen. Liegt bereits ein Ereignis vor, wird dieses durch den Konsumenten verbraucht, und er kann seinen Rechenprozeß fortsetzen.
Dieses prinzipielle Verhalten der Systemdienste des Event-Objektes wurde im EMOS dahingehend erweitert, daß jedes Event 16 Einzelereignisse speichern kann, die durch den Dienst *signalisieren* gesetzt werden und durch den Dienst *erwarten* einzeln oder mehrfach unter Angabe logischer Verknüpfungen ausgewertet werden können.
In den Kernen EBS, IRTS und BOS 1810 muß die Ereignissynchronisation unter Benutzung des im ursprünglichen Sinn für die Betiebsmittelverwaltung definierten Semaphorobjektes erfolgen. Zu diesem Zweck ist der Semaphor mit dem Anfangswert Null zu definieren. Die Ereignissynchronisation erfordert dann vom Produzenten die Benutzung des Dienstes *freigeben* und vom Konsumenten die Benutzung des Dienstes *anfordern* des Semaphorobjektes.

Die Verwaltung von Betriebsmitteln

Die zur Verwaltung von Betriebsmitteln in Echtzeitkernen implementierten Mechanismen können eingeteilt werden in
* Verwaltung von Einzelbetriebsmitteln
* Verwaltung von Betriebsmittel-Pools.

Exclusiv nutzbare Einzelbetriebsmittel erfordern einen wechselseitigen Zugriff der mit ihnen arbeitenden Rechenprozesse. Die Programmsequenz eines Rechenprozesses, für die dieser das unteilbare Recht zur Benutzung des Betriebsmittels erhalten muß, wird als kritischer Abschnitt bezeichnet.
Die Kerne EBS und IRTS gestatten durch eine Einklammerung des kritischen Abschnittes in die Systemdienste *Scheduler aus* und *Scheduler ein* eine einfache Methode der Betriebsmittelverwaltung. Nachteilig dabei ist aber, daß für die Dauer der Betriebsmittelbenutzung alle Rechenprozesse, also auch jene, die das Betriebsmittel nicht benötigen, von der Konkurrenz um die CPU ausgeschlossen werden.
Durch die Anwendung des Semaphoreobjektes ist dieser Nachteil behoben. Bei der Verwaltung von Einzelbetriebsmitteln wird der Semaphore mit einem Anfangswert von eins initialisiert.
Durch den Systemdienst *anfordern* kann ein Rechenprozeß für den Fall, daß der Semaphore eins beträgt, sofort das Betriebsmittel erhalten. Der Semaphorewert wird dekrementiert, und der Rechenprozeß kann weiterarbeiten. Ist das Betriebsmittel bereits vergeben, und der Semaphorewert ist kleiner eins, wird der Wert ebenfalls dekrementiert und das Task-Objekt des den Dienst benutzenden Rechenprozesses in die Warteschlange des Semaphors eingekettet. Damit wird die Task dem Zustand *gestoppt* zugeordnet.
Ein nicht mehr benötigtes Betriebsmittel muß am Ende des kritischen Abschnitts durch den Systemdienst *freigeben* an den Semaphor zurückgegeben werden. Dieser Dienst inkrementiert den Semaphorwert unter Beachtung der oberen Grenze von eins. Wird dabei die Obergrenze erreicht, kann der Rechenprozeß seine Arbeit fortsetzen. Liegt ein anderer Semaphorewert vor, geht die Benutzung des kritischen Abschnitts an die im Kopf der Semaphorwarteschlange eingekettete Task über. In diesem Fall werden die ausgekettete und die den Dienst *freigeben* benutzende Task in den Zustand *bereit* eingeordnet, und es wird zum Scheduler verzweigt.
Semaphore können dynamisch (EBS, IRTS, BOS 1810) und während der Generierung des Kerns statisch (IRTS, EMOS) definiert werden. Die Verwaltung der Warteschlange liegt systemspezifisch fest (EBS, EMOS = Priorität, IRTS = FIFO) oder kann bei Semaphordefinition (BOS 1810 = Priorität oder FIFO) bestimmt werden. Darüber hinaus gestatten EBS und BOS 1810 die Angabe einer Wartezeit im Dienst *anfordern*. Ist nach deren Ablauf die Benutzung des kritischen Abschnitts nicht gewährleistet, wird die Task aus der Warteschlange entfernt und in den Zustand *bereit* eingeordnet.
Bei der Anwendung des Semaphores treten dann Probleme auf, wenn Rechenprozesse, die den kritischen Abschnitt besitzen, suspendiert oder gelöscht werden oder hochpriorisierte Rechenprozesse übermäßig lange auf die Freigabe eines durch einen niederpriorisierten Rechenprozeß belegten Abschnitts warten müssen.
Diese Nachteile werden durch das Region-Objekt ausgeschlossen, das sich vom Semaphor nur hinsichtlich eines erweiterten Algorithmus der gleichen Systemdienste unterscheidet. So verhindert das Region-Objekt das Löschen oder Suspendieren des Task-Objektes, dessen Rechenprozeß den kritischen Abschnitt benutzt. Weiterhin wird dieser Rechenprozeß für den Fall, daß eine höherpriorisierte Task in der Regionwarteschlange auf Zuweisung wartet, bis zur Freigabe des kritischen Abschnitts mit der Priorität der wartenden Task abgearbeitet. Die Kerne EBS, EMOS und BOS 1810 enthalten diesen Mechanismus, der für den Nutzer ein effektives Mittel zur Implementierung eines wechselseitigen Ausschlusses bietet.
Die Kerne EBS, IRTS und BOS 1810 unterstützen auf der Grundlage der Definition mehrwertiger Semaphore die Verwaltung von Betriebsmittel-Pools. Durch die Benutzung des Dienstes *anfordern* erhält der Rechenprozeß das Recht zur Nutzung eines (EBS, IRTS) oder mehrerer (BOS 1810) Betriebsmittel des Pools. Hierbei muß angemerkt werden, daß durch den Semaphor nur die Rechte zur Benutzung, aber keine Zeiger auf die konkret vorgesehenen Betriebsmittel verwaltet werden.
Für die Verwaltung des Betriebsmittels Speicher stellen alle hier betrachteten Kerne spezielle Systemdienste bereit. Unter Angabe der gewünschten Größe wird durch den Dienst *anfordern Speicher* ein Zeiger auf ein Segment des Speicherpools übergeben. Falls dieses Segment nicht mehr benötigt wird, können Rechenprozesse mittels des Dienstes *Speicher freigeben* das Segment an den Pool zurückgeben.

Nachrichtenkommunikation

Die Kommunikation dient vorrangig dem Austausch von Nachrichten zwischen Rechenprozessen und beinhaltet gleichzeitig eine Form der Synchronisation. Es werden die Verfahren
* synchroner Nachrichtenaustausch
* asynchroner Nachrichtenaustausch

unterschieden. Darauf aufbauend können Nachrichten sowohl in Form der vollständigen Nachricht, bezeichnet als *message passing*, als auch in Form eines Zeigers auf die Nachricht, bezeichnet als *message token passing*, zwischen Rechenprozessen ausgetauscht werden. In beiden Verfahren findet das Mailbox-Objekt Anwendung. Eine Übersicht über die nutzbaren Verfahren der Echtzeitkerne EBS, IRTS, EMOS und BOS 1810 zeigt Tafel 3.
Der synchrone Nachrichtenaustausch erfolgt ohne Zwischenspeicherung der Nachricht. Sende- und Empfangsprozeß melden ihren Kommunikationswunsch über einen Systemdienst der Mailbox beim Kern an. Dieser kettet die Task, die jeweils als erste den Dienst benutzt, in den Zustand *gestoppt* ein. Wird daraufhin der Kommunikationsdienst durch den Partnerrechenprozeß ausgelöst, führt der Kern den Nachrichtenaustausch durch, ordnet beide Tasks in den Zustand *bereit* ein und verzweigt zum Scheduler. Damit bedingt der synchrone Nachrichtenaustausch eine starke Kopplung der beteiligten Rechenprozesse.
Der asynchrone Nachrichtenaustausch führt zu einer Entkopplung zwischen den Sende- und Empfangsprozessen. Hier übernimmt die Mailbox die Verwaltung der auf eine Nachricht wartenden Tasks sowie der für einen Empfänger abgelegten Nachrichten.
Der Systemdienst *senden* kann eine Nachricht in einer Mailbox ablegen, ohne den Laufzustand des Rechenprozesses zu beeinflussen. Das erfolgt dann, wenn noch Platz in der Box ist und kein Empfänger bereits auf die Nachricht wartet. Ist die Box gefüllt, wird die Sendetaks in die Warteschlange der Mailbox

Tafel 3 **Kommunikationsverfahren**

Betriebs-system	EBS	IRTS	EMOS	BOS 1810
Verfahren				
* Protokoll	synchron/ asynchron	asynchron	synchron/ asynchron	asynchron
* Festlegung Protokoll	bei Definition des Objektes	standard	im Systemdienst Senden	standard
[illegible] verwaltun				
* Definition	dynamisch	statisch dynamisch	statisch	dynamisch
* Warteschlange für Tasks				
** Strategie ** time-out	Priorität vorhanden	Priorität vorhanden	Priorität nicht vorhanden	Priorität vorhanden
* Warteschlange für Nachrichten				
** Strategie	FIFO/ LIFO	FIFO	Priorität	FIFO/ Priorität
** Festlegung Strategie	im Systemdienst Senden	standard	im Systemdienst Senden	bei Definition des Objektes
** Verwaltungseinheit	Festlegung bei Definition des Objektes; von 0 bis 32765 Byte	Message = Linkefeld + 26 Byte Steuerinfo + Nachricht	3 Worte = * Nachricht oder * Adresse und Länge der Nachricht	2 Worte = * Objekttoken Nachricht * Objekttoken für Quittung

eingekettet und dem Zustand *gestoppt* zugeordnet.
Wartet bereits ein Empfänger, erhält dieser sofort die Nachricht, beide Rechenprozesse werden in *bereit* eingeordnet, und der Scheduler entscheidet, welcher Rechenprozeß fortschreiten darf.
Der Systemdienst *empfangen* kann nur eine von bereits vorhandenen Nachrichten aus der Mailbox entnehmen, ohne den Laufzustand des Rechenprozesses zu beeinflussen. Ist keine Nachricht vorhanden, wird die Empfangstask in die Warteschlange eingekettet und der Rechenprozeß dem Zustand *gestoppt* zugeordnet. EBS, IRTS und BOS 1810 gestatten ein zeitlich befristetes Warten auf den Empfang einer Nachricht. Die Task des diese Option benutzende Rechenprozesses wird nach Ablauf dieser Zeit (time out) unabhängig vom Vorhandensein einer Nachricht nach *bereit* verlagert.
Das Betriebssystem EBS ermöglicht durch spezielle Generierungsparameter des Kommunikations-Objektes (Anzahl und Länge der Nachrichtenpuffer) alle Verfahren des Nachrichtenaustausches. Die implementierten Systemdienste erlauben eine einfache und übersichtliche Handhabung.
Den von der Anzahl der Systemdienste her umfangreichsten Service bietet IRTS. Es benutzt neben der Mailbox eine als Message bezeichnete Datenstruktur, die außer der eigentlichen Nachricht weitere Steuerinformationen beinhaltet. Diese Informationen gestatten eine umfassende Synchronisation zwischen Sender und Empfänger. Insbesondere gestattet IRTS in den Diensten *senden* und *empfangen* das Adressieren der Nachricht. Somit werden in den IRTS-Mailboxen gleichzeitig Nachrichten und Tasks für den Fall verwaltet, daß ihre Adressen nicht übereinstimmen.
EMOS unterstützt sowohl synchronen wie asynchronen Nachrichtenaustausch. Die Auswahl des Verfahrens erfolgt durch Parameter der Systemdienste *senden* und *empfangen.*
Das System BOS 1810 kommt mit vier Systemdiensten aus. Der Systemdienst *senden* hinterlegt in der Mailbox zwei Zeiger. Ein Zeiger ist ein Verweis auf die Nachricht. Der andere zeigt auf ein Objekt, welches zur Synchronisation zwischen Empfänger und Sender genutzt werden kann. Diese Synchronisation ist insbesondere bei message token passing erforderlich, um ein erneutes Benutzen des Nachrichtenpuffers durch den Sender vor Abschluß der Verarbeitung der Information durch den Empfänger auszuschließen.

Ausblick

Aus der vorliegenden Analyse ausgewählter Funktionsprinzipien der Echtzeitkerne EBS, IRTS, EMOS und BOS 1810 kann nicht die Auswahl eines Kerns für einen konkreten Einsatzfall abgeleitet werden. Dazu sind weit umfangreichere und gegenüber herkömmlichen Betriebssystemen auch andersartige Vergleichskriterien heranzuziehen. Wesentliche Elemente dieses Vergleiches sollten sein:

* Weitere Funktionsprinzipien wie
  - Zeitverwaltung
  - Behandlung von Ausnahmebedingungen
* Softwareunterstützung hinsichtlich
  - Generierbarkeit
  - Interfaces zu Hochsprachen
  - Inbetriebnahme und Testhilfsmittel
* Anzahl und Komplexität der bereitgestellten Systemdienste und der von ihnen benötigte Zeitanteil
* Verfügbarkeit für Prozessortyp, Speicherplatzbedarf, PROM-Fähigkeit.

Die vorliegende Arbeit bietet hierbei einen Ansatz durch den Vergleich ausgewählter Funktionsmodule der aus meiner Sicht in der DDR praxiswirksamen 16-Bit-Echtzeitbetriebssysteme.

Literatur

/1/ Fritzsch, W.: Prozeßrechentechnik. Berlin: VEB Verlag Technik 1986
/2/ iRMX86 – Reference Manual. Intel Corporation, Santa Clara 1980
/3/ iRMX286 – Reference Manual. Intel Corporation, Santa Clara 1985
/4/ Graf, M.: RMOS2/286 – ein modernes Echtzeitbetriebssystem. mini Micro magazin, (1987) 9, S. 42
/5/ Krapp, M.; Zerbe, V.; Jahn, D.: Systemprogrammierung in Mudula-2. unveröffentlichtes Manuskript
/6/ Krapp, M.; Richter, J.; Schwartz, J.: Eine FORTH-Systemfamilie. Mikroprozessortechnik, Berlin 2 (1988) 2, S. 53
/7/ Kompf, R.: Echtzeitbetriebssystem EBS 8001 – Anwendungsbeschreibung. Institut für Energetik, Leipzig 1985
/8/ Bala, P.; Haupt, R.; Claßen, L.: Echtzeitbetriebssystem IRTS 8000. Mikroprozessortechnik, Berlin 1 (1987) 1, S. 8
/9/ Anwendungsbeschreibung IRTS-Kern. Zentrum für Forschung und Technologie des KEAW, Berlin 1987
/10/ Anwenderhandbuch – ICA Software. Zentrum für Forschung und Technologie des KEAB, Berlin 1988
/11/ BOS 1810 – Systemunterlagendokumentation. VEB Robotron-Projekt Dresden, 11/86
/12/ Giloi, W. K.: Rechnerarchitektur. Berlin, Heidelberg, New York, Springer Verlag: 1981
/13/ Horn, T.: Zur Gestaltung, Implementierung und Analyse von Betriebssystemen für Klein- und Mikrorechner. Dissertation B, Technischen Universität Dresden 1983
/14/ Bunata, T.; Huber, B.: 16-Bit-Mikroprozessor für Echtzeit-Multitask-Betrieb. Elektronik, München 32 (1983) 7, S. 53
/15/ Product overview – the transputer family. INMOS Group of Companies, June 1986

Eingabe paralleler Prozeßdaten

Dr. Reinhard Friedemann, Eckhard Johne, Dr. Franz Röller
Hochschule für Verkehrswesen „Friedrich List" Dresden, Sektion Nachrichtentechnik

Zur Kopplung von Prozessen an Mikrorechner werden oft parallele Schnittstellen benötigt. Sind diese (wie an den meisten Personalcomputern) nicht vorgesehen, ist zur Nachrüstung ein Eingriff in das Gerät selbst notwendig. Im Beitrag wird eine Möglichkeit beschrieben, durch die mit relativ geringem Aufwand und ohne Veränderungen im Rechner eine Reihe von Anwendungsfällen lösbar werden.

Grundgedanke

Die V.24 als Standardschnittstelle ist für die Ein-/Ausgabe serieller Informationen vorgesehen. Ihre Nutzung als Prozeßeingabe zum Beispiel von einem parallelen Druckausgang eines digitalen Meßgerätes, setzt eine Umwandlung parallel vorliegender Informationen in serielle Daten voraus. Zusätzlich ist eine Anpassung an das Datenformat (Start- und Stoppschritte) sowie ein relativ genauer Takt erforderlich. Zur Steuerung der Übertragungseinrichtung (Modem) verfügt die Schnittstelle jedoch über einige weitere, statisch wirkende Signalleitungen. Durch geschickte Ausnutzung dieser läßt sich eine Eingabeschaltung aufbauen, die mit wenigen Bauelementen eine parallele Dateneingabe ermöglicht.
Die für eine Pegelanpassung TTL–V.24 notwendigen zustätzlichen positiven und negativen Spannungen (5...15 V) können infolge des großen Toleranzbereiches durch Ausnutzung der Signalleitungen aus der V.24 selbst gewonnen werden.

Schaltungsbeschreibung

Entsprechend Bild 1 gelangen die Prozeßdaten über die parallelen Eingänge der Multiplexer (D1 und D2), eine ODER-Schaltung (Diode V1, V2), und die Pegelanpassung (D3) auf die Signalleitung 106 (CTS) der Schnittstelle. Dieses Signal kann am SIO-Schaltkreis der Schnittstelle nach dem Statusreset im Register 0 als Bit 5 abgefragt werden. Die Steuerung der Multiplexer erfolgt über die V.24-Signale 105 (RTS) und 108 (DTR) mittels NAND-Schaltung (D5) und

Zähler (D4). Zur Pegelanpassung sind entsprechende Widerstände (R4, R5) sowie Schutzdioden (V3, V4) vorgesehen. Das statische Ausgangssignal der Leitung 105 (RTS), im Schreibregister 5 das Bit 1, dient zur Auswahl der Daten von Multiplexer 1 oder 2. Mit der L/H-Flanke dieses Signals wird gleichzeitig der Zähler (D4) weitergeschaltet und damit das folgende Eingangleitungspaar adressiert. Nach 16 Flanken sind alle 32 Eingangsleitungen genau einmal abgefragt worden.

Zum Rücksetzen dieses Zählers dient ein kurzer positiver Impuls auf der Leitung 108 (DTR), im Schreibregister 5 das Bit 7. Damit wird ein definierter Anfangszustand zu Beginn der Abfrage erreicht. Der negative Pegel der Leitung 108 dient gleichzeitig als Stromversorgung der Pegelanpassung (D3).

Die Sendedatenleitung 103 führt ständig positiven Pegel und liefert die fehlende Speisespannung für die Pegelanpassung. Zusätzlich kann an diese Leitung ein Piezoschwinger zur Ausgabe eines akustischen Signals angeschlossen werden. Dazu ist das Bit *Send Break* (im Schreibregister 5 das Bit 4) im Rhythmus der Tonfrequenz alternierend zu setzen und zu löschen.

Beispielprogramm

Zum besseren Verständnis der Schaltung werden zwei Beispielprogramme für Dateneingabe sowie Tonausgabe angegeben (Bild 2). Im Ergebnis der Dateneingabe werden die 32 Bit zu je vier Bit (entspricht einer BCD-Stelle) in einem Integer-Feld abgelegt. Die weitere Verarbeitung (z. B. Entprellung durch Mehrfachabfrage u. s. w.) ist vom jeweiligen Anwendungsfall abhängig.

✉ KONTAKT ☏

Hochschule für Verkehrswesen „Friedrich List" Dresden, Sektion Nachrichtentechnik, WB 14130, PSF 103, Dresden, 8072; Tel. 33 62

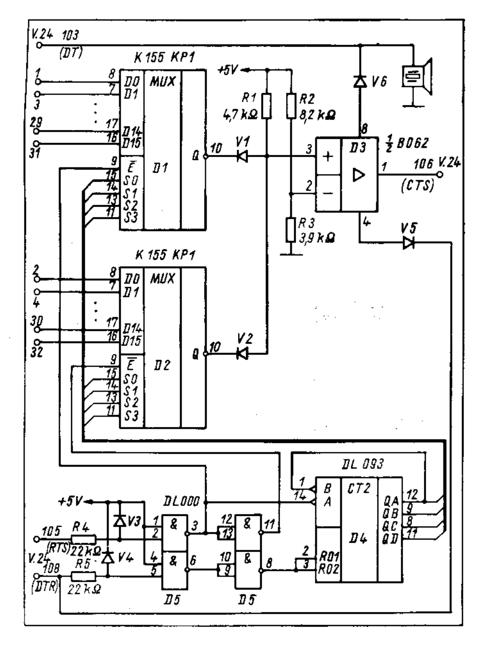


Bild 1

```
PROCEDURE TON(dauer,hoehe:integer);
var i,j,merker:integer;

{Piezoschwinger liegt auf Datenleitung}

BEGIN
 merker:=$12;                        {RTS+,Daten+}
 for i:=dauer downto Ø do
 begin
  for j:=Ø to hoehe do begin end;    {Verzoegerung}
  Port[tor]:=5;{WR5}
  Port[tor]:=merker;
  merker:=merker xor $1Ø;            {alternierende Ausgabe +/-}
 end;
 Port[tor]:=5;                       {Zugriff WR 5 vorbereiten}
 Port[tor]:=$12;            {Normalzustand Daten +, fuer Speisung Pegelanpassung}
END;

{Testprogramm}

var wert:Eingangsdaten;
    i   :integer;

{$U+}
BEGIN
repeat
 Eingabe(wert);
 for i:=Ø to 7 do
  write(wert[i]:2);
 writeln;
 Ton(1ØØ,5);
until keypressed;
END.
```

```
PROGRAM BEISPIELEINGABE;

const tor=$f;{$f  Toradresse beim PC 1715 (SIO Steuerport)
              $51      -"-    -"- BC A512Ø}

type  Eingangsdaten=array[Ø..7] of integer;

 PROCEDURE EINGABE(var wert:Eingangsdaten);

 var i,j:integer;

{$U-}
 BEGIN
  Port[tor]:=Ø;Port[tor]:=Ø;        {Leerbefehle}
  Port[tor]:=5;                     {Zugriff WR 5 vorbereiten}
  Port[tor]:=$92;                   {Reset Zaehler}
  for i:=Ø to 7 do wert[i]:=Ø;      {Zeit fuer Reset Zaehler am Adapter nutzen}
  Port[tor]:=5;
  Port[tor]:=$12;        {DTR auf -, Daten auf +, RTS auf + damit MUX 1 aktiv}
  for i:=Ø to 7 do                  {Einlesen von acht BCD-Werten}
  begin
   j:=Ø;

   while j<4 do                     {Einlesen eines BCD-Wertes}
   begin
    Port[tor]:=$1Ø;                 {Statusreset zum Einlesen CTS}
    if (Port[tor] and $2Ø)=Ø then wert[i]:=wert[i]+(1 shl j);{Biteinbau}
                         {^ Eingabe von CTS (bit 5) }
    Port[tor]:=$15;                 {Statusreset+WR5 aktivieren}
    Port[tor]:=$1Ø;               {RTS auf -, MUX 2 fuer ungerade bit aktivieren}
    j:=j+1;
    Port[tor]:=$1Ø;                 {Statusreset zum Einlesen CTS}
    if (Port[tor] and $2Ø)=Ø then wert[i]:=wert[i]+(1 shl j);
    Port[tor]:=$15;
    Port[tor]:=$12;                 {MUX 1 aktiv, Zaehler weiterstellen}
    j:=j+1;
   end;{while}

  end;{for}
 END;{Eingabe}
```

Bild 2
Beispielprogramme

Nutzerspezifische Gerätetreiber unter MS-DOS

Dr. Martin Reinhardt
Friedrich-Schiller-Universität Jena, Sektion Technologie für den wissenschaftlichen Gerätebau

Für verschiedene Anwendungsfälle ist es wünschenswert, eigene Geräte an 16-Bit-Personalcomputer anzuschließen. Dabei sollte man dafür sorgen, daß die Programme, die diese Geräte bedienen, sich entsprechend den Konventionen des verwendeten Betriebssystems verhalten. Personalcomputer, die mit dem Betriebssystem MS-DOS arbeiten, besitzen in der Regel mindestens eine serielle Schnittstelle. Diese kann allerdings meist nur mit einem Hardwareprotokoll entsprechend den V.24-Konventionen betrieben werden. Viele in der DDR produzierte Geräte mit V.24 Interface (z. B. Drucker) benötigen dagegen oft das DC1/DC3-(XON/XOFF-)Softwareprotokoll. Es ist demzufolge notwendig, den Treiber der seriellen Schnittstelle durch einen anderen zu ersetzen.

Im Betriebssystem MS-DOS werden block- und zeichenorientierte Gerätetreiber voneinander unterschieden. Blockorientierte Treiber werden für Disketten- oder Festplattenlaufwerke benötigt, zeichenorientierte zum Beispiel für Drucker, Tastatur und Bildschirm.

Der Standardgerätetreiber für die serielle Schnittstelle wird unter dem Namen COM1: angesprochen. Er ist wie alle anderen Gerätetreiber in einer verketteten Liste im Speicher abgelegt. Dabei wird die Verkettung über einen Zeiger im Kopf des Treibers, der auf den nächsten Treiber weist, erreicht.

Für den Aufruf des Treibers erwartet das Betriebssystem zwei Eintrittspunkte. Der erste ist für spätere Multitasking-Umgebungen vorgesehen. Da diese durch das Betriebssystem bis zur Version 3.3 nicht zur Verfügung gestellt wird, hat die über diesen Eintrittspunkt erreichbare Routine nur eine Aufgabe. Sie legt einen bei Treiberaufruf vom Betriebssystem in den Registern ES und BX übergebenen Zeiger in einem treiberinternen Speicher ab. Dieser Zeiger weist auf eine Liste (Request Header), die alle Informationen für die Abarbeitung des Treibers enthält (z. B. Kommandokode entsprechend dem auszu-

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;13. 8. 88 M. R.
;Treiber für V.24
;setzt int 14h auf einen DC1-DC3 Treiber
;Copyright by Friedrich-Schiller-Universität Jena
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Vereinbarungsteil
;
dc1 = 11h
dc3 = 13h
dc4 = 14h
;*** offsets der Request-Header-Variablen ***
cmd     = 2             ;Kommandos-Code
status = 3              ;Gerätestatuswort, Eintritt = 0
                        ;Austritt = Fehlercode + Zustand
trans   = 14            ;Transferadresse für Treiberende
;*** Fehlermeldungen bzw. Status ***
d$error    = 10000000b  ;allg. Fehler
d$done     = 00000001b  ;Funktion ausgeführt
d$cmd$err = 3           ;ungült. Kommandocode
;
int14nr    = 14h
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Kopf des Treibers
xonxoff     segment
        assume cs:xonxoff, ds:xonxoff
            dw -1,-1                ;kein weiterer Treiber
            dw 1000000000000000b    ;Bit 15 zeichenorientiert
            dw multitask            ;Multitasking Eintritt
            dw entrypoint           ;Treibereintritt - Kommandoentschl.
            db "DC1/DC3 "           ;Treibername
;Tabelle der Treiberfunktionen
tabjmp      dw init                 ;V.24 Initialisierung int14h-Verbieg.
            dw cmd$err              ;keine weiteren Funktionen im-
            dw cmd$err              ;plementiert
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
            dw cmd$err
tabend:
;4 Byte zum Ablegen der Anfangsadresse des Request Headers durch
;die Multitask-Routine
rh$ptr      dd 0                    ;Request-Header-Pointer
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Multitask-eintritt Request-Header-Pointer ablegen
entry       proc far
multitask:
            mov word ptr cs:[rh$ptr],bx     ;Request-Header-P.
            mov word ptr cs:[rh$ptr+2],es   ;ablegen
            ret
entry       endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Eintrittspunkt des Treibers, Kommandoauswertung
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
entrypoint:

            push si
            push ax
            push cx
            push dx
            push di
            push bp
            push ds
            push es
            push bx
            lds bx,cs:[rh$ptr]          ;ds:bx - Request Header
            mov al,byte ptr ds:[bx].cmd ;al::=Kommandocode, cmd =
                                        ;offset zum request header Anfang
            cmp al,(tabend-tabjmp)/2
            jge cmd$err                 ;Code ungültig
            mov si,offset tabjmp        ;Anfangsadresse Adresstabelle
            add si,ax
            add si,ax                   ;si::=Handleradresse
            les  di,dword ptr ds:[bx].trans
                                        ;es:di::=Transferadresse
            push cs
            pop ds                      ;Datensegment::=codesegment für si
            jmp word ptr[si]            ;jmp ds:si zum treiberanfang
;Fehlerroutine
cmd$err:    mov al,d$cmd$err            ;Kommandofehler
            mov ah,d$error or d$done
            jmp short ex1
;Treiberende, Register herstellen
dev$exp     proc far
dev$ex:     mov ah,d$done               ;ok
ex1:        lds bx,cs:[rh$ptr]
            mov word ptr [bx].status,ax ;Gerätestatus
            pop bx
            pop es
            pop ds
            pop bp
            pop di
            pop dx
            pop cx
            pop ax
            pop si
            ret
dev$exp     endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;DC1/DC3 Treiber wird neue int 14H Routine
;Input:  ah = Funktionsnummer
;        al = zu sendendes Zeichen
;Output: ah = Status
;        al = empfangenes Zeichen
;Kill:   ax
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
v24onoffp  proc far
v24onoff:  push dx
           cmp ah,0
           jz initial
           cmp ah,1
           jz senden
           cmp ah,2
           jz empfangen
           cmp ah,3
           jz status_abfr
           jmp short v24ende
;
;Empfangen eines Zeichens
;input: -
;output: al=zeichen, ah bit 7=1 - time-out kein Zeichen empfangen
;
empfangen:  push cx                     ;Zähler für time-out
            mov cx,0fffh
            mov dx,3fdh
char_nook:  in al,dx
            and al,01
            jnz char_ok
            loop char_nook
            mov ah,80h                  ;time out
            jmp short empfend
char_ok:    mov dx,3f8h
            in al,dx                    ;Dateneingabe
empfend:    pop cx
            jmp short v24ende
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Initialisierung und Statusabfrage nicht implementiert
;Kill:  ax
initial:
status_abfr:
            mov ax,6000H
            jmp short v24ende
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Senden eines Zeichens mit DC1/DC3-Protokoll
;input:  al=zu sendendes Zeichen
;output: ah=bit 4=1 d. h. DC4 empfangen
senden:     call charout
            call charin
            and al,7fh
            cmp al,0
            jz sendenend        ;kein Echo vom Empfänger
            cmp al,dc1
            jz sendenend        ;weitere Ausgaben ok
            cmp al,dc3
            jnz dc4test
senden1:    call charin         ;Empfänger gesperrt, auf Freigabe
;warten
            and al,7fh
            cmp al,dc1
            jnz senden1
            jmp short sendenend
dc4test:    cmp dc4
            jnz sendenend
            mov ah,10h                  ;bit 4 gesetzt = dc4 erkannt
            jmp short sendenend
sendenend:  jmp short v24ende
v24ende:    pop dx
            iret
v24onoffp   endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
char        proc near
;Zeichen an 8250 ausgeben, Zeichen in al

;input:      al=auszugebendes Zeichen
;output:     -
;kill:       dx
charout:     push ax
forever:     mov dx,3fdh                ;e/a- Adresse Leitungsstatusreg.
             in al,dx
             test al,20h                ;Senderegister leer ?
             jz forever
             pop ax
             mov dx,3f8h                ;e/a - Adresse Senderegister
             out dx,al
             ret

;Zeichen empfangen, kein Zeichen al=0
;input:      -
;output:     al=Zeichen
;kill:       dx
charin:      mov dx,3fdh
             in al,dx
             test al,1
             jz nochar
             mov dx,3f8h                ;Adr. Empfängerregister
             in al,dx
             jmp short charine
nochar:      mov al,0
charine:     ret
char         endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Treiberinitialisierung, neue int 14h Prozedur eintragen
init:       mov ax,offset v24onoff
            mov bx,0
            push ds
            mov ds,bx
            mov bx,50h
            mov [bx],ax                 ;Interruptvektor änd.
            push cs
            pop ax
            inc bx
            inc bx
            mov [bx],ax
            pop ds
            call uart
            mov dx,offset text
            mov ah,09h                  ;Stringausgabe
            int 21h
;Adresse des ersten freien Bytes des Treibers übergeben da
;Initialisierung weg kann
            lds bx,cs:[rh$ptr]
            mov word ptr [bx].trans,offset init
            mov [bx].trans+2,cs
            jmp dev$ex
text:       dw 0d0ah
            db "DC1/DC3 Treiber installiert"
            dw 0d0ah
            db "$"
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;initialisiere den uart 8250 mit:
;8 daten-bits,1 stopbit,ohne Parität und 9600 baud
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
uart:       mov     dx,3fbh     ;e/a-Adresse des Leitungssteuerregisters
            mov     al,80h
            out     dx,al       ;zur Adressierung der Baudraten-register
            mov     dx,3f8h     ;e/a-Adresse der Baudraten-register(lsb)
            mov     al,0ch      ;lsb-Wert für 9600 baud
            out     dx,al
            mov     dx,3f9h     ;e/a-Adresse der Baudraten-register(msb)
            mov     al,0        ;msb-Wert für 9600 baud
            out     dx,al
;Baud-rate ist initialisiert
```

```
;initialisiere nun Leitungssteuerregister
          mov     dx,3fbh     ;e/a-Adresse des Leitungssteuerregisters
          mov     al,03h      ;ohne Parität,1 Stoppbit,8 Datenbits
          out     dx,al
          mov     dx,3fch     ;e/a-Adresse des Modemsteuerregisters
          mov     al,00       ;
          out     dx,al
;sperre alle 4 Arten von Unterbrechungen
          mov     dx,3f9h     ;e/a-Adresse des Unterbrechungs-
                              ;aktivierunsreg.
          mov     al,0
          out     dx,al
          ret
;Initialisierung des 8250 beendet
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
xonxoff   ends
          end
```

führenden Kommando und Gerätestatus für die Rückgabe von Fehlermeldungen). Der zweite Treibereintrittspunkt weist auf einen Programmteil, der die eigentliche Kommandoauswertung vornimmt und die geforderten Aktionen ausführt. Dazu wird aus dem Request Header der Kommandokode entnommen, eine entsprechende Entschlüsselung vorgenommen, das Kommando ausgeführt und im Gerätestatus die Ausführung quittiert oder eine Fehlermeldung eingetragen.

Für den Kopf des Treibers ergibt sich folgender Aufbau:

32 Bit	Zeiger auf die Anfangsadresse des nächsten Treibers (= OFFFFH, wenn in der gleichen Datei kein weiterer Treiber folgt, Verkettung realisiert das Kommando Device in der CONFIG. SYS-Datei)
16 Bit	Treibercharakteristik (Bit 15 = 1 – zeichenorientierter Treiber)
16 Bit	Offset des Multitaskingeintrittspunktes des Treibers
16 Bit	Offset der Kommandoentschlüsselung und -verarbeitung des Treibers
8 Byte	Treibername (mit Leerzeichen auffüllen)

Im Anschluß an diesen Kopf folgt eine Tabelle, die die Offsets aller vom Treiber ausführbaren Kommandos enthält (maximal 24).

Danach schließen sich Multitaskingeintritt, Kommandoentschlüsselung und -verarbeitung an.
Die in MS-DOS vorhandenen Treiber greifen zur Initialisierung und zur Bedienung der Schnittstellen auf den BIOS-ROM des jeweiligen Rechners zu. Das BIOS enthält die hardwareabhängigen Bestandteile des Betriebssystems für den jeweiligen Rechner. Der Aufruf dieser Routine erfolgt über Softwareinterrupts, wobei zum Beispiel für die serielle Schnittstelle der Interrupt 14H genutzt wird /1/.
Leider arbeiten viele Standardprogramme unter MS-DOS so, daß sie direkt die BIOS-Routinen aufrufen. Von solchen Programmen wird ein neuer Gerätetreiber nicht erreicht, da er nur über die logische Schnittstelle des Betriebssystems (DOS-Aufrufe) angesprochen werden kann. Aus diesem Grund wurde für das vorliegende Beispiel für die serielle Schnittstelle kein vollständig neuer Treiber mit allen notwendigen Funktionen geschrieben, sondern der dargestellte Treiber enthält nur den Initialisierungsteil. Alle anderen Kommandos führen zu einer Fehlermeldung.
Der im Programm enthaltene Initialisierungsteil verändert nur den für die serielle Schnittstelle zuständigen Interruptvektor (14H). Der neue Vektor zeigt auf die notwendige DC1/DC3-Prozedur, die ebenfalls in dem dargestellten Programm enthalten ist. Diese neue INT-14H-Routine ermöglicht die Ausgabe von Zeichen an die serielle Schnittstelle mittels Softwareprotokolls. Die Statusabfrage und die Einstellung der Parameter (Bitrate, Parität und Anzahl der Stoppbits) sind in dem dargestellten Beispiel nicht implementiert, können aber leicht hinzugefügt werden.
Um möglichst wenig Speicherplatz zu verbrauchen bietet MS-DOS die Möglichkeit, den Speicherplatz, den der Initialisierungsteil des Treibers benötigt, nach erfolgter Initialisierung wieder frei zu geben. Dazu muß zum einen die Initialisierung am Ende des Treibers angeordnet werden, und zum anderen muß man bei der Initialisierung einen 32-Bit-Zeiger ab Speicherplatz *Request Header + 14* ablegen. Dieser Zeiger gibt den ersten freien Speicherplatz (Offset, Segment) nach dem installierten Treiber an.
Das vorliegende Programm wird mit Hilfe des Device-Befehls in der CONFIG. SYS-Datei installiert. Dazu muß eine Datei vom Typ .exe erst übersetzt und in eine Datei vom Typ .bin umgewandelt werden. Nun wird die .bin-Datei noch in das ROOT-Directory der Boot-Diskette (Festplatte) kopiert und der Befehl
Device = Name.bin
in die Datei CONFIG. SYS eingetragen. Nach dem nächsten Booten des Betriebssystems ist der Treiber installiert, und ein Gerät, das mit Softwareprotokoll arbeitet, kann bedient werden.

Literatur

/1/ Schmidt, H.-J.: IBM-BIOS-Funktionen. mc (1986) 6, S. 40
/2/ Esders, E.: Treiberimplementation in MS-DOS. mc (1987) 12, S. 132

✉ KONTAKT ☏
Friedrich-Schiller-Universität Jena, Sektion Technologie, Technikum LAURA, Ernst-Thälmann-Ring 32, Jena, 6900; Tel. 8 22 21 32

Hardwarenahe Programmierung in C

Dr. Uwe Schneider, Detlef Bauer
VEB Werkzeugmaschinenkombinat „FRITZ HECKERT" Karl-Marx-Stadt, Forschungszentrum des Werkzeugmaschinenbaues

Serielle Ein-/Ausgabe

Zur Kommunikation mit der Peripherie können unter MS-DOS die seriellen Schnittstellen COM1 und COM2 genutzt werden. Die Programmbeispiele in den Bildern 1 und 2 zeigen einige Möglichkeiten, wie das aus C heraus möglich ist. Sie demonstrieren auch die Verwendung von Interrupts in C.
Zur Initialisierung der Schnittstelle sowie zur ungepufferten Ein-/Ausgabe kann der Schnittstelleninterrupt 14H verwendet werden. Unterfunktion 0 dient der Initialisierung, wie in init_com ⟨⟩ dargestellt. Die Parameter bedeuten: nr: 0 = COM1, 1 = COM2, bd: Bitrate, stop: Stoppbits, bit: Wortlänge. Mit Unterfunktion 1 werden Zeichen gesendet. Der Output-Status im AH-Register kann durch status_com ⟨⟩ ausgewertet werden. out_com ⟨⟩ erwartet einen Pointer auf den auszugebenden Datenbereich sowie die Anzahl der Bytes. Den Zeichenempfang ermöglicht Unterfunktion 2. in_com ⟨⟩ fragt zusätzlich die Tastatur ab, um mit ∧C abbrechen zu können, wenn 00 empfangen wird.
Während mit den CP/M-ähnlichen Funktionen 3 und 4 des INT 20H nur COM1 angesprochen wird, kann mit den XENIX-kompatiblen File-Management-Funktionen 3FH und 40H des INT 20H jedes Gerät und jede Datei angesprochen werden. Dazu ist zunächst ein logischer Zugriffskanal einzurichten ⟨create_handle⟨⟩⟩. Anschließend kann mit write_device ⟨⟩, read_device ⟨⟩ blockweise aus- und eingegeben werden, indem ein Pointer auf den Speicherbereich und die Anzahl zu übertragender Bytes angegeben werden. In gleicher Weise arbeiten die E/A-Funktionen des untersten Niveaus von C: open ⟨⟩, write ⟨⟩ und read ⟨⟩.

Schnelle Bildschirmausgabe

Menüs stellen eine effektive Software-Anwenderschnittstelle dar. Die grafischen Möglichkeiten des erweiterten IBM-Zeichensatzes erlauben bereits anspruchsvolle Lösungen. Störend wirkt, daß der Bildaufbau bei der Verwendung des printf ⟨⟩-Befehls sehr langsam erfolgt. Durch direktes Schreiben in den Bildschirmpuffer des Rechners läßt sich dieser Nachteil beheben. Die sich daraus ergebenden Kompatibilitätsprobleme können mit den dargestellten Funktionen gelöst werden.
Durch Beschreiben des Bildschirmpuffers und Lesen des Inhalts ⟨Attribut, Code⟩ wird getestet, ob der Rechner XT-kompatibel ist ⟨Monochromadapter⟩. Durch den Attributtest kann dabei erfaßt werden, ob der ANSI. SYS-Treiber installiert wurde. In Abhängigkeit vom Testergebnis wird ein Zeiger auf eine der beiden Funktionen putnorm ⟨⟩ oder putscreen ⟨⟩ eingerichtet. putscreen ⟨⟩ schreibt direkt in den Bildschirmpuffer, putnorm führt printf ⟨⟩ aus.
Diese Ausgabefunktion ist für Programme gedacht, die vorwiegend auf XT-kompatiblen Rechnern laufen, aber auch auf anderen Rechnern lauffähig sein sollen. Auf die beschriebene Weise kann natürlich auch auf andere Monitoradapter getestet werden.

```
#include <string.h>
#include <stdio.h>
#include <dos.h>
#define Ctrl_C  03
union REGS inr,outr;
struct SREGS inx;

/* Schnittstelleninitialisierung INT 14h Typ 0                  */
/* Defaultwerte: 9600 Bd, No Parity, 1 Stoppbit, 8 Bit          */
/* nr - Schnittstellennr. (COM1=0, COM2=1)                      */
init_com(nr,bd,par,stop,bit)
int nr,bd,stop,bit;
char par;
{ inr.x.dx=nr;
  inr.h.ah=0;
  switch(bd)
  { case  110: inr.h.al=0; break;
    case  150: inr.h.al=32; break;
    case  300: inr.h.al=64; break;
    case  600: inr.h.al=96; break;
    case 1200: inr.h.al=128; break;
    case 2400: inr.h.al=160; break;
    case 4800: inr.h.al=192; break;
    default:   inr.h.al=224; break;
  }
  if(par=='O'||par=='o') inr.h.al+=8;
  if(par=='E'||par=='e') inr.h.al+=24;
  if(stop==2) inr.h.al+=4;
  switch(bit)
  { case 5: break;
    case 6: inr.h.al++; break;
    case 7: inr.h.al+=2; break;
    default: inr.h.al+=3; break;
  }
  int86(0x14,&inr,&outr);
}

/* Eingabe von COMnr, wenn empfangener Wert gleich 00, dann  */
/* Tastaturabfrage; return( Ctrl C oder ASCII-Code)           */
in_com(nr)
int nr;
{ inp: inr.x.dx=nr;
       inr.h.ah=2;
       int86(0x14,&inr,&outr);
       if(outr.h.al!=0 && outr.h.ah==0)
         { stat_com(outr.h.ah);
           return(outr.h.al); }
       inr.h.ah=0xb;
       intdos(&inr,&outr); /* Tastaturstatus */
       if(outr.h.al)
         { inr.h.ah=1;
           intdos(&inr,&outr); /* Zeichen holen */
           if(outr.h.al==Ctrl_C)
             return(outr.h.al); }
       goto inp;
}

/* Ausgabe eines anz-langen Bytepuffers, adressiert ueber */
/*  ptr, an COMnr  */
out_com(nr,ptr,anz)
int nr,anz;
char *ptr;
{ int i;
  while(anz--)
   { inr.h.ah=1;
     inr.x.dx=nr;
     inr.h.al=*ptr++;
     int86(0x14,&inr,&outr);
     stat_com(outr.h.ah);
   }
}

/*  Auswertung des Sende/Empfangsstatus */
stat_com(reg)
int reg;
{ switch(reg)
    { case 1: printf("\noverrun error"); break;
      case 2: printf("\nparity error"); break;
      case 4: printf("\nbreak recieved"); break;
      case 5: printf("\nTX-holding-register empty"); break;
      case 6: printf("\nTX-shift-register empty"); break;
      case 7: printf("\ntimeout error"); break;
    }
}

/* create handle oeffnet logischen Kanal der Zugriffsart        */
/* zg; der device name steht ab segment:offset                  */
create_handle(seg,off,zg)
int seg,off,zg;
{ inr.h.ah=0x3c;
  inr.x.cx=zg;
  inr.x.dx=off;
  inx.ds=seg;
  intdosx(&inr,&outr,&inx);
  return(outr.x.ax);
}

/* write device gibt an Kanal handle anz-Zeichen aus, die       */
/* ab outseg:outoff stehen;. return() gibt tatsaechliche        */
/* Anzahl gesendeter char zurueck                               */
write_device(handle,anz,outseg,outoff)
int handle,anz,outseg,outoff;
{ inr.h.ah=0x40;
  inr.x.bx=handle;
  inr.x.cx=anz;
  inr.x.dx=outoff;
  inx.ds=outseg;
  intdosx(&inr,&outr,&inx);
  return(outr.x.ax);
}

/* read device lieszt vom Kanal handle anz-Zeichen und          */
/* speichert sie ab inseg:inoff; return() gibt Anzahl           */
/* tatsaechlich gelesener char zurueck                          */
read_device(handle,anz,inseg,inoff)
int handle,anz,inseg,inoff;
{ inr.h.ah=0x3f;
  inr.x.bx=handle;
  inr.x.cx=anz;
  inr.x.dx=inoff;
  inx.ds=inseg;
  intdosx(&inr,&outr,&inx);
  return(outr.x.ax);
}

/* Beispielprogramm                                             */
char name[80],device[]="COM2",*nptr,*dptr;
main()
{ int j,bytes,handle,wanz;
  nptr=name; dptr=device;
  init_com(1,4800,'E',2,7);
  while((name[j]=in_com(1))!='\r')
    if((name[j++]==Ctrl_C)||(j>80)) exit(1);
  name[j]='\0';
  bytes=strlen(name);
  handle=create_handle(FP_SEG(dptr),FP_OFF(dptr),0);
  wanz=write_device(handle,bytes,FP_SEG(nptr),FP_OFF(nptr));
    if(wanz!=bytes) exit(1);
}
```

Bild 1 Serielle Ein-/Ausgabe

```
#include <stdio.h>

#define SCI       printf("\033[1m")               /* intensiv       */
#define SCR       printf("\033[7m")               /* invers         */
#define SCN       printf("\033[0m")               /* normal         */
#define POS(x,y)  printf("\033[%d;%dH",x,y)       /* Cursor auf x,y */
#define VRAM      0xb0000000                      /* Video-RAM Adr. */

int (*out)()=NULL; /* Zeiger auf Funktion                           */

/* testet, ob sich Bildschirmpuffer beschreiben laeszt ______ */
test_xt()
  { int putnorm(),putscreen();
    int far *testptr=(int far*)VRAM;
    POS(1,1); SCN; printf("X");
    if (*testptr != 0x0758)
       { out=putnorm; return(0); }
    out=putscreen;
    return(1);
  }

/*  Schreibe bei XT direkt in den Bildschirmpuffer __________ */
putscreen (zei,spa,str,attr)
int zei,spa,attr;
register unsigned char *str;
  { char far*sp=(char far*)VRAM;
    register unsigned char far *bp=sp+2*80*zei+2*spa;
    while(*str != 0)
      { *bp++=*str++;
        *bp++=attr;
      }
  }

/* printf(), wenn nicht XT _________________________________ */
putnorm (zei,spa,str,attr)
int zei,spa,attr;
register unsigned char *str;
  { switch(attr)
      { case 0x0f: SCI; break;
        case 0x70: SCR; break;
        default  : SCN;
      }
    POS(++zei,++spa);
    printf("%s",str);
    SCN;
  }

/* Beispiel (*out)(zeile,spalte,string,attribut);                   */
main() {
test_xt();
(*out)(4,0,"TESTPROGRAMM: AUSGABE MIT (*OUT)()",0x7);
      }
```

Bild 2 Schnelle Bildschirmausgabe

✉ KONTAKT ☏

Forschungszentrum des Werkzeugmaschinenbaus Karl-Marx-Stadt im Werkzeugmaschinenkombinat „Fritz Heckert", Abt. 219, PSF 1061, Karl-Marx-Stadt, 9010; Tel. 59 94 35.

TERMINE

Fachtagung „Mikrorechner und ASICs in der Meß- und Automatisierungstechnik"

WER? Fachausschuß Mikroprozessor-Interface-Systeme der wissenschaftlichen Sektion Computertechnik der KDT und Technische Universität „Otto von Guericke" Magdeburg, Sektion Automatisierungstechnik und Elektrotechnik
WANN? 27. bis 28. März 1990
WO? Magdeburg
WAS?
- Meßwerterfassung mit Mikrorechnern
- Kommunikations- und Mikrorechentechnik
- kundenspezifische Schaltkreise

WIE? Teilnahmewünsche bitte an: Kammer der Technik, Tagungsbüro, Brüderstraße 3, Stendal 1, 3500
Inhaltliche Rückfragen an: Technische Universität „Otto von Guericke", Sektion Automatisierungstechnik und Elektrotechnik, Doz. Dr. sc. techn. Michaelis, Postschließfach 124, Magdeburg, 3010

Prof. Dr. sc. Seifart

Große Programmsysteme in C auf 8-Bit-Rechnern

Dr. Berndt Götze, Falk Ambos
Technische Universität Dresden, Sektion Informationstechnik

Der nachfolgende Beitrag soll die Erfahrungen, die bei der Programmierung komplexer Programmsysteme in der Sprache C gewonnen wurden, darstellen. Dabei sollen besonders die Aspekte der Nutzung der effizienzerhöhenden Sprachkonstruktionen und die Modularisierbarkeit umfangreicher Programmsysteme im Vordergrund stehen. Gelegentlich wird aber auch auf einige pragmatische Aspekte bezüglich der existierenden verschiedenen C-Compiler hingewiesen.

Erfahrungsbasis

Als Ausgangspunkt für die folgenden Darlegungen dienen Erfahrungen, die im wesentlichen bei der Implementierung des Logikanalysatorbetriebssystems LABS86-PC gesammelt wurden. Dieses Programmsystem ist eine Weiterentwicklung des Betriebssystems LABS83 /1/ für den Logikanalysator LA32/20. Wie bereits in /1/ beschrieben, stellt das Logikanalysatorbetriebssystem LABS83 ein sehr leistungsfähiges Softwarepaket dar. Es ist vollständig menügesteuert und bietet dem Nutzer unter dem Aspekt der Programmierung eines Meßgerätes zur Logikanalyse einen Komfort, der vergleichbar mit dem von Softwarelösungen entsprechender Systeme führender ausländischer Hersteller ist.
LABS86-PC enthält Verbesserungen und Erweiterungen, die vor allem der Bereitstellung von grafischen Auswertemöglichkeiten der Meßergebnisse und der Unterstützung der Analogsignalaufzeichnung dienen /2/. Es wurde für den Anschluß eines vom Zentrum für wissenschaftlichen Gerätebau und Forschungstechnik der TU Dresden entwickelten Transientenvorsatzes vorbereitet und durch Bereitstellung von spezieller Hardware (Grafisches Farbdisplay GFD1520 /3/ und /4/) und Programmerweiterungen mit grafischen Auswertemöglichkeiten für die verarbeiteten analogen Signalverläufe ausgestattet. Die erreichten Parameter des Systems zeigt Tafel 1.

Tafel 1 Technische Parameter des LA32/20 mit LABS86-PC

Kanalzahl	32 digitale oder 4 Analogkanäle
maximale Abtastfrequenz	1 MHz
nutzbare Abtastfrequenzen im LA32/20	1; 5; 10; 25; 50 kHz 100; 250; 500; 1 000 kHz (2,5; 5; 10; 20 MHz bei paralleler Digitalaufzeichnung)
externer Abtasttakt	≦ 1 MHz
Speichertiefe/Analogkanal	2 048 oder 4 × 512 Ereignisse
Auflösung des ADU	8 Bit
Meßbereiche der Eingangsspannung	0 bis 100 mV 0 bis 200 mV 0 bis 500 mV 0 bis 1 V 0 bis 2 V 0 bis 5 V 0 bis 10 V
programmierbare Eingangsoffestspannung	0 bis 100 % in 10 % Schritten bezogen auf den Meßbereich
Triggerung analog	einer der vier Analogeingänge oder externer Triggereingang
Triggerung digital	alle Triggermöglichkeiten des LA32/20

Das Betriebssystem wurde für den Personalcomputer PC 1715, ergänzt durch ein Beistellgefäßsystem BG 1715 zur Logikanalyse, entwickelt. Dabei wurden die in PL/M80 vorliegenden Quelltexte von LABS83 für die Programmierung hinzugezogen. Die Umprogrammierung der Basissoftware erfolgte auch aus Gründen der Loslösung von den wenigen zur Verfügung stehenden Rechnern mit PL/M80-Compilern. Es muß jedoch betont werden, daß sich der vorgenommene Umstieg von PL/M80 auf C nicht in jedem Falle als Gewinn erwies. Die entsprechenden Gründe dafür sind im wesentlichen in der fehlenden Systemunterstützung von großen in C kodierten Programmsystemen auf CP/M-kompatibeln Betriebssystemen zu suchen. Unbestreitbar ist diese Tatsache, wenn man die Geschlossenheit und die Einheit der Soft- und Hardware sowie die vorausschauende Bereitstellung von geeigneten Mitteln zur Programmierung von Mikrorechnern durch die Firma Intel berücksichtigt. Außerdem bietet PL/M80 nahezu die gleichen Leistungsparameter wie die UNIX-Programmiersprache C und ist auf den Rechnersystemen, auf denen sie verfügbar ist, zumeist optimal an die vorhandene Prozessorarchitektur angepaßt.
Die vorliegende Version des Programmsystems hat zur Zeit einen Umfang von zirka 8 000 Quellzeilen (in ungefähr 75 Modulen) und wurde mit einer Überlagerungsstruktur entwickelt. Auf die dabei aufgetretenen Probleme wird im folgenden noch näher eingegangen.

Struktur des Programmsystems

Das Programmsystem gliedert sich entsprechend den bereits erwähnten Überlagerungsstrukturen in vier komplexe Teilprogramme. In der Wurzel des Systems (Root) sind das Rahmenprogramm, das die zentrale Steuerschleife repräsentiert, sowie die Verwaltung der Überlagerungsdateien und andere systemglobale Dienste enthalten. Die anderen drei Teilprogrammsysteme weisen aufgabenspezifische Überlagerungsstrukturen auf (Bild 1).

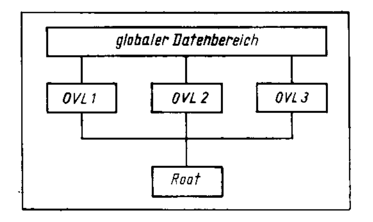


Bild 1 Gesamtstruktur des Programmsystems

Für das Programmsystem wurde eine feste Hauptspeicheraufteilung gewählt. Sie gliedert sich in separate Bereiche für das Wurzelprogramm, die Überlagerungen, den globalen Datenbereich und in eine für spezielle Daten verwendete Speicherfläche (Bild 2).
Alle verwendeten Überlagerungsdateien werden beim Start des Programmsystems geöffnet, wodurch ein stets effizienter Zugriff durch den Überlagerungsverwalter auf den benötigten Lademodul erfolgt. Diese Tatsache und die Konsequenz, daß der ursprünglich in C entwickelte Lader für Überlagerungsdateien in Assemblersprache umkodiert wurde, resultieren hauptsächlich aus den Restriktionen, die dem Programmierer auf 8-Bit-Systemen herkömmlicher Art auferlegt sind. Jedoch soll damit nicht die Untauglichkeit von Rechnern der 8-Bit-Klasse nachgewiesen werden. Wir sind prinzipiell der Ansicht, daß eine große Anzahl der Problemlösungen effizient und ohne „Rückfall" in die umständliche Assemblerprogrammierung auf herkömmlichen 8-Bit-Systemen erreicht werden kann.
Aufgrund des Fehlens von flexiblen Linkern für Dateien mit Überlagerungsstrukturen in den bekannten CP/M-kompatiblen Betriebssystemen machte sich die Verwendung einer Überlagerungsdatei, die für den globalen Datenbereich des gesamten Betriebssystems steht, erforderlich. Diese Datei enthält im wesentlichen vorinitialisierte Daten sowie Adressen von Funktionen der Wurzel, die

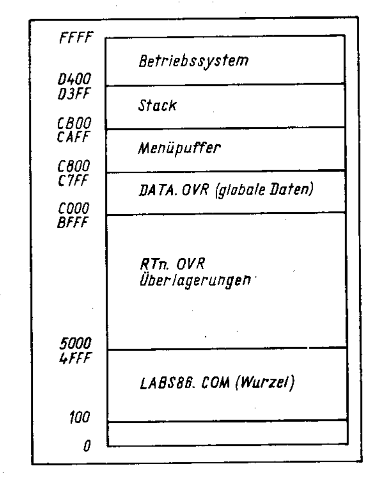


Bild 2 Hauptspeicheraufteilung des Programmsystems

```
/* globaler Datenbereich (Ausschnitt) */

/* Funktionen der Wurzel, die systemglobal verwendet werden */

extern int display();
extern unsigned char scp(),
                     access(),
                     bideh();

/* Synonyme fuer die systemglobalen Funktionen
 * der Wurzel, die in den Ueberlagerungen verwendet werden.
 * Die entstehenden ungeloesten Referenzen beim Linken des
 * globalen Datenbereiches an die Ueberlagerungen aufgrund
 * des Fehlens der oben deklarierten Funktionen, sind nicht
 * relevant, da als globaler Datenbereich der Datenmodul des
 * Wurzelprogramms verwendet wird.
 */

int (*displ)() = display;
unsigned char (*scpl)() = scp,
              (*accessl)() = access,
              (*bidehl)() = bideh;
```

Bild 3 Ausschnitt des globalen Datenbereiches

global für alle Überlagerungen zur Verfügung stehen sollen. Das Problem der Übergabe dieser Funktionsadressen an die Überlagerungen konnte durch einen Trick mit in die Linkinformation hineingebracht werden (Bild 3). Absolute Adressen mußten lediglich zur Festlegung der Ladeadresse für die Überlagerung und deren Einsprungadresse, sowie der Ladeadresse des globalen Datenbereiches verwendet werden.
Diese sicher etwas zweifelhafte Vorgehensweise bei der globalen Nutzung von Wurzelfunktionen löst jedoch zunächst einige grundlegende Probleme bei der Programmierung von Überlagerungen und ist viel mehr Ausdruck der unzureichenden Eigenschaften der CP/M-kompatiblen Betriebssysteme als Entwicklungssystem und der Dissonanzen in der vorhandenen Entwicklungssoftware.

Implementierung des Betriebssystems

Für die Implementierung des Betriebssystems in den einzelnen Moduln wurde der C-Compiler von Robotron CC-1520 verwendet. Er stellt dem Benutzer, bis auf wenige Ausnahmen, den gesamten Sprachumfang der Sprache C zur Verfügung und bietet deshalb wesentliche Vorteile gegenüber anderen Compilerimplementierungen auf 8-Bit CP/M-Systemen (SuperSoft, BDS-C, Aztec-C). Die Kodegenerierung erfolgt in der Assemblersprache des Prozessors Intel8080, dessen Maschinenbefehle eine Untermenge der des U 880 Prozessors sind. Wie Erfahrungen aus der Programmierung großer Assemblerprogramme zeigen, ist die Generierung von Intel-8080-Assemblerkode kein Nachteil, da die beim U 880 zur Verfügung stehenden zusätzlichen Befehle nur einen geringen Anteil der Assemblerprogramme bilden.
Als nachteilig bei der Verwendung des Compilers CC-1520 müssen jedoch die lange Übersetzungszeit sowie die eingeschränkten Möglichkeiten bei der Verwendung externer Datenbereiche in den Programmen aufgrund des großen Speicherumfangs der einzelnen Compilerläufe genannt werden. Um diesem Nachteil entgegenzuwirken, wurde ein CP/M-kompatibles Betriebssystem mit einem nutzbaren Speicherbereich von 54 KByte generiert.
Bei der Kodierung der Quellprogramme wurden fast alle effizienzerhöhenden und bis in das Maschinenniveau herunterreichenden Sprachelemente von C verwendet. So kamen unter anderem sehr häufig Ausdrücke mit Formen der bedingten Bewertung und umfangreiche Bitstrukturen zur Anwendung. Dabei erwiesen sich bedingte Bewertungen aufgrund ihrer Universalität als sehr nützlich, vor allem in komplexen Entscheidungsausdrücken. Hingegen konnten mit Bitstrukturen nur bedingt gute Erfahrungen gesammelt werden. Zwar läßt sich durch die Vergabe von Namen für einzelne Bits eine bessere Lesbarkeit der Programme erzielen, jedoch steht dem der große Aufwand der Zusammenfassung einzelner Bits zu Bytes oder Wörtern und der vom Compiler erzeugte sehr große Kodeumfang gegenüber. Es wurden Erkenntnisse gesammelt, die zeigen, daß der Compiler spezielle Masken zur Veränderung der Bits in den Strukturen generiert und außerdem besondere Funktionen zu deren Manipulation gerufen werden müssen.

Literatur

/1/ Götze, B.; Nicklisch, G.; Meusel, K.-H.: Logikanalysator LA32/20. Radio, Ferns. Elektron., Berlin 34 (1985) 10, S. 626 und 34 (1985) 11, S. 719
/2/ Meusel, K.-H.; Wissmann, F.; Ambos, F.: Analogaufzeichnung mit Logikanalysator. Nachrichtentechnik, Elektronik, Berlin 37 (1987) 4, S. 143
/3/ Hartmann, M.; Hiller, H.; Meusel, K.-H.: Modulares Farbgrafiksystem GFD 1520/m und seine Anwendung. Radio, Ferns. Elektron., Berlin 36 (1987) 2, S. 71
/4/ Götze, B.; Hartmann, M.; Meusel, K.-H.: Software zum modularen Farbgrafiksystem GFD 1520/m. Radio, Ferns. Elektron., Berlin 36 (1987) 2, S. 218

✉ KONTAKT ☏

Technische Universität Dresden, Sektion Informationstechnik, Mommsenstraße 13, Dresden, 8027; Tel.: 4 63 21 51/ 5 92 34 94

Wegbereiter der Informatik

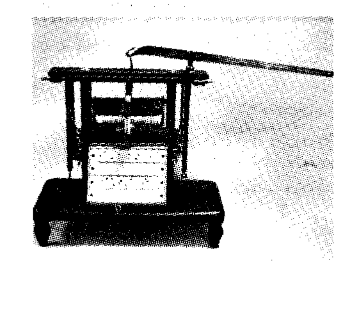


HERMANN HOLLERITH

∗ 1860 Buffalo
† 1929 bei Washington

Die Geschichte der Informationsverarbeitung ist untrennbar mit Herrmann Hollerith, dem Begründer der Lochkartentechnik, verbunden. Die Erfindung, Konstruktion und Anwendung seiner Lochkartenmaschinen stellt einen wesentlichen Markstein in der Gesamtentwicklung der Computertechnik dar, war doch zu Beginn die Lochkarte der einzige maschinenlesbare Datenträger.
Die Lochkarte war zu seiner Zeit prinzipiell schon bekannt. Ähnlich wie sein Landsmann Falcon (1728) hatte der französische Mechaniker Jaques de Vaucanson (1709–1782) im Jahre 1745 für das Abheben der Kettfäden in Webstühlen einen „Programmspeicher" in Gestalt einer umlaufenden Blechwalze mit Lochungen gebaut, um ein Webmuster selbsttätig wiederholen zu können. Der französische Weber Joseph-Marie Jacquard (1752–1834) verbesserte diese Erfindung 1805, indem er ein Lochbandprogramm aus Kartonkarten daraus entwickelte und dies zur Steuerung eines Webstuhls verwendete. Unser Bild zeigt ein Verkaufsmodell des Jacquardschen Webstuhls aus dem Jahre 1810.
Über Holleriths Lebenslauf ist leider nicht allzuviel bekannt. Seine Eltern – beide Pfälzer Abstammung – wanderten 1848 nach Amerika aus; sein Vater war bis dahin Studienrat am Gymnasium in Speyer. Nach dem Schulbesuch studierte Hollerith an der Bergakademie der Universität Columbia und legte dort bereits mit 19 Jahren seine Diplomprüfung ab. Er blieb zunächst als Assistent bei seinem Lehrer und war mit Fragen der Industriestatistik beschäftigt. Im Jahre 1882 ging er an das Massachusetts Institute of Technology und wurde Lehrer für Technische Mechanik. Ein Jahr später übersiedelte er nach St. Louis, wo er elektromagnetische Bremsen für das Eisenbahnwesen entwickelte. Außerdem arbeitete er 1884–1889 für das Patentamt in Washington. In dieser Zeit reifte auch Holleriths Erfindung. Er entwickelte eine Lochkartenmaschine (von ihm übrigens *Tabulating Machine* bzw. *Tabulator* genannt), die aus einem Kartenlocher, einem elektromagnetischen Zähler und einer Sortiereinrichtung bestand. Das von ihm verwendete Lochkartenformat, das dem einer 20-Dollar-Note entsprach, und der von ihm benutzte Zahlencode sind bis heute im Gebrauch. Zum Abtasten der Lochkarte ersann Hollerith metallische Fühlstifte, über die elektrische Kontakte geschlossen werden können. Seine Maschinen, die damals bis zu 1 000 Karten in der Stunde verarbeiten konnten, brachten in Baltimore und New York solchen Erfolg, daß sich das „Bureau of the Census" entschloß, die 11. amerikanische Volkszählung 1890 mit Hollerithmaschinen auszuwerten. Dadurch konnte die Zählung nach reichlich 1 Jahr abgeschlossen werden – die vorangegangene Volkszählung hatte noch über 7 Jahre gedauert! Bald darauf wurden seine Maschinen auch in Kanada, Norwegen und Österreich eingesetzt. Hollerith reiste sogar nach Rußland, um die Organisation der dortigen Volkszählung persönlich mit vorzubereiten. Seit 1895 standen Hollerithmaschinen auch in Berlin und Paris.
Im Jahre 1896 gründete Hollerith die *Tabulating Machine Company;* sie produzierte sowohl die Maschinen als auch die Lochkarten. Aus dieser Firma ging später die bekannte IBM-Corporation hervor; Hollerith war bis 1921 beratender Ingenieur dieser Gesellschaft. Für seine Maschinen, die er ständig verbesserte, hat Hollerith mehrere Patente erhalten: sein erstes 1884, sein letztes 1919. Im Jahre 1890 wurde er von der Universität Columbia zum Ehrendoktor ernannt.
In der DDR wurden Lochkartenmaschinen (die Soemtron nach dem Hollerithprinzip fertigte) bis in die 60er Jahre verwendet, zum Beispiel für Abrechnungsprojekte. Diese Maschinen konnten nicht nur Zähl- und Sortiervorgänge, sondern auch einfache arithmetische Operationen erledigen.
Heute haben die programmgesteuerten EDV-Anlagen die Lochkartenmaschinen zumeist ersetzt, allenfalls werden die Lochkarten noch zur Dateneingabe verwendet. Dieser Umstand schmälert jedoch Holleriths Erfindung in keiner Weise; vielmehr hat er durch seine Pionierarbeit für den heutigen Entwicklungsstand der Computertechnik entscheidend den Weg mit bereitet.

Dr. Klaus Biener



Entwurf von Gate-Array-Schaltkreisen Teil 4

Prof. Dr. Dietmar Müller
Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik

Zusammenarbeit von Anwender und Hersteller

Die Zusammenarbeit von (Anwender-)Entwerfer und Hersteller bedingt bei Gate-Array-Schaltkreisen ein hohes Maß an Kooperation. Es wurde bereits mehrfach ausgeführt, daß der Hersteller letztlich Schaltkreise präpariert, von denen er keine genaue Kenntnis hat. Dies erfordert:

- klare Schnittstellen vom Anwender zum Hersteller
  und
- klare Schnittstellen vom Hersteller zum Anwender.

Diese Schnittstellen müssen sowohl entwurfstechnologische als auch vertragsrechtliche und damit wirtschaftliche Aspekte beinhalten.
Für die Zusammenarbeit sind verschiedene Varianten möglich, die sich vor allem durch den Umfang der Arbeit des Anwenders und der verfügbaren Rechentechnik unterscheiden. Auf der Basis des Entwurfsablaufes des Bildes 2 im Teil 1 /1/ sind im Bild 13 mögliche Varianten der notwendigen Kooperation dargestellt.

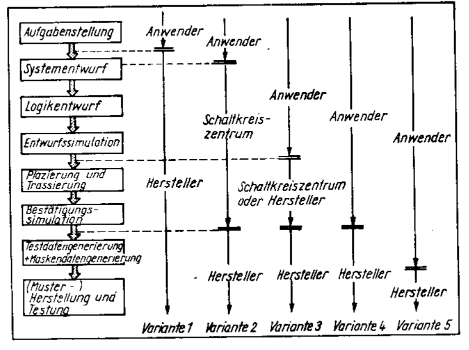


Bild 13 Varianten der Zusammenarbeit zwischen Anwender und Hersteller

Variante 1 ist für Gate Array-Schaltkreise völlig untypisch. Sie war und ist charakteristisch für den Entwurf und die Herstellung von Vollkundenwunsch-Schaltkreisen. Der Anwender – in diesem Falle der Auftraggeber – übergibt nur die Aufgabenstellung und eventuell noch Informationen zur Systementwicklung. Mit diesen Informationen entwirft und produziert der Hersteller in voller Verantwortung für das Produkt. Der Entwurf kann bei hohem Aufwand optimal durchgeführt werden, was zu kleiner Chipfläche und damit hoher Ausbeute führt. Diese Variante ist prinzipiell nur für kundenspezifische Schaltkreise mit sehr hohen Stückzahlen sinnvoll. Nach der Definition von ASIC /2/ gehört ein deratig entworfener Vollkundenwunsch-Schaltkreis nicht zur Klasse der ASICs.

Variante 4 entspricht dem Vorgehen eines Anwenders, der alle notwendigen hard- und softwareseitigen Voraussetzungen zum Gate-Array-Entwurf besitzt; neben dem kompletten CAD-System bedeutet dies eine leistungsfähige 32-Bit-Rechentechnik (K 1840) vor allem für die Plazierung und die Trassierung. Da wegen des CAD-Systems die Rechentechnik von Anwender und Hersteller zumindest kompatibel sein muß, ist es unbedeutend, ob der Anwender auch die Testvektoren zur Strukturprüfung berechnet – wobei er dem Hersteller ein Magnetband mit Steuerdaten für den Patterngenerator und mit Testdaten für den Tester übergibt – und dadurch der Hersteller zur „Silizium-Schmiede" (*Variante 5*) wird, oder ob der Anwender, wie bei Variante 4, mittels Entwurfsfreigabe die Verantwortung für den Entwurf ebenfalls vollständig übernimmt und der Hersteller die Test- und Layoutdaten quasi als „Dienstleistung" berechnet.

Variante 2 stellt eine Lösung für Anwender dar, denen die notwendigen technischen Voraussetzungen fehlen oder die auch perspektivisch nur wenige Typen von Gate-Array-Schaltkreisen benötigen. Dabei würde in Form einer Dienstleistung ein Entwurfszentrum eines Anwenders der Variante 4 oder 5 oder ein territoriales, gemeinsam zu nutzendes Entwurfs- bzw. Schaltkreiszentrum (z. B. Schaltkreiszentrum des VEB Textimaelektronik Karl-Marx-Stadt) den eigentlichen Entwurf als Auftragsentwurf durchführen. In Anbetracht von Sinn und Zweck der Gate-Arrays sollten aber auch diese Anwender bemüht sein, *ihre* Gate-Array-Schaltkreise selbst zu entwerfen, wenn es die Bearbeitungskapazitäten zulassen. Sie sollten dazu den Beratungs- und Informationsdienst (einschließlich der Entwurfserfahrungen) sowie die Entwurfstechnik und die Entwurfssysteme der entsprechenden Schaltkreiszentren (kostenpflichtig) nutzen, um mit ihrem Systemwissen und ohne Zeitverlust durch den Wechsel der Bearbeiter effektiv zu arbeiten. Diese Zusammenarbeit und insbesondere die Mittlerfunktion der Schaltkreiszentren zwischen Anwender und Hersteller wird später noch genauer dargestellt.

Variante 3 als Mischform verdient Aufmerksamkeit. Der Anwender besitzt eine relativ leistungsfähige PC-Technik (z. B. EC 1834, A 7100 oder XT-kompatibel). Auf dieser Technik sind Teile des industriell nutzbaren Gate-Array-Entwurfssystems verfügbar. Mit seinen Komponenten (Compiler für Beschreibungssprache, Simulator u. a.) ist der Anwender in der Lage, seine „Schaltung" rechnerunterstützt zu entwerfen. Er kann am Arbeitsplatz die Funktion des zukünftigen Schaltkreises mit der Entwurfssimulation überprüfen, er kann im Dialog Schaltungsänderungen ausführen, und er erhält qualitative Aussagen, ob seine entworfene Struktur die geforderte Sollfunktion erfüllt. Dies ist ihm möglich, ohne daß eine bis ins letzte Detail formulierte und schriftlich fixierte Aufgabenstellung vorliegt. Der Rechner dient als echte Unterstützung des kreativen Entwerfers.

Nach erfolgreichem Abschluß der Entwurfssimulation kann die Schaltungsbeschreibung als vollständig geprüfter NBS-Quelltext problemlos auf die 32-Bit-Rechentechnik der Anwender nach Variante 4 oder 5 auf die eines Schaltkreiszentrums übernommen werden. Dabei ist es sinnvoll, daß der bisherige Bearbeiter seinen Gate-Array-Schaltkreis weiter „betreut". Obwohl der eigentliche kreative Entwurf abgeschlossen ist, kann die Bestätigungssimulation wiederum am besten vom Entwerfer durchgeführt werden.
Mit den Systemen ARCHIMEDES (VEB Forschungszentrum Mikroelektronik Dresden) und PC-GAD (Technische Universität Karl-Marx-Stadt) ist eine derartige Variante beim System U 5200 möglich (siehe Teil 2 /3/). Für das System U 5300 wird es künftig ebenfalls möglich sein.

Zusammenarbeit von Anwender und Schaltkreiszentrum

Die Varianten 2 und 3 im Bild 13 zeigen die Mittlerrolle eines Schaltkreiszentrums zwischen Anwender und Hersteller. Im Bild 14 (nach /4/) ist dies genauer dargestellt. Es ist die aktive Mitwirkung des Schaltkreiszentrums bei beiden Entwurfsarten zu erkennen. Die umfangreichen (akkumulierten) Entwurfserfahrungen eines Schaltkreiszentrums gestatten eine fundierte Beratung bereits bei der Beantwortung der Frage nach der Entwurfsart.

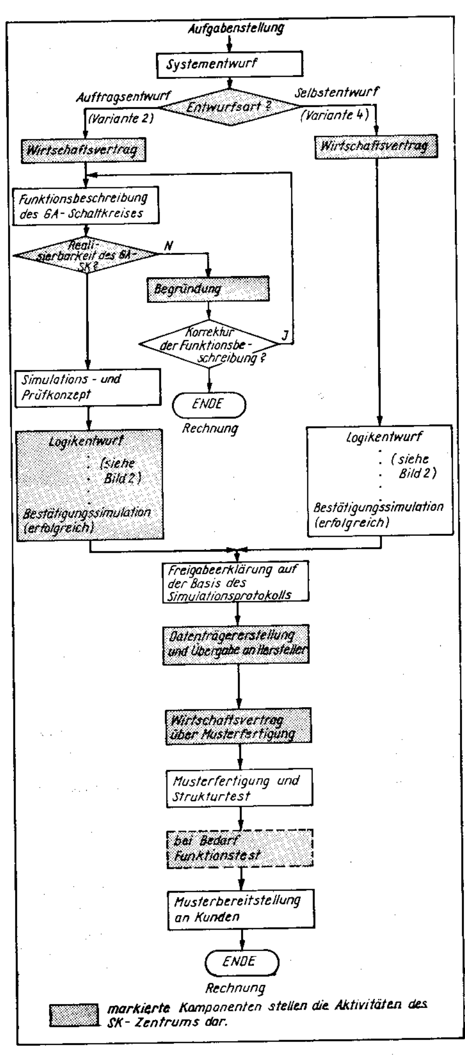


Bild 14 Varianten der Zusammenarbeit zwischen Anwender und Schaltkreiszentrum

Entscheiden sich die Partner für einen **Selbstentwurf** durch den künftigen Anwender, werden in einem Wirtschaftsvertrag die konkreten Betreuungsleistungen, die kostenpflichtige Nutzung des CAD-Systems einschließlich der Rechentechnik sowie die Entwurfstermine und die Termine für die Musterbereitstellung fixiert. Der Abschluß der konkreten Entwurfsarbeit des Anwenders erfolgt durch die Bestätigungssimulation (siehe Teil 1 in MP 3/1989). Das Protokoll dieser abschließenden Simulation wird durch die Unterschrift des Anwenders zum Freigabedokument des Entwurfs.

Auch beim **Auftragsentwurf** werden in einem Wirtschaftsvertrag die Leistungen und die Termine des Schaltkreiszentrums und des Anwenders fixiert. Der Anwender muß eine Funktionsbeschreibung des künftigen Gate-Array-Schaltkreises übergeben. Da die Nutzung einer Funktionsbeschreibungssprache noch nicht – wie bei der Strukturbeschreibung mit NBS – festgelegt worden ist, kann die Beschreibung in einer der vielfältigen und bekannten Formen erfolgen (von Booleschen Gleichungen bis zur verbalen Beschreibung; jedoch sollte ein fixierter TTL-Logikplan tunlichst nicht Ausgangspunkt eines Gate-Array-Entwurfs sein). Wie Bild 14 zeigt, wird anschließend die Realisierbarkeit der Vorgaben geprüft.

Bei Nichtrealisierbarkeit werden dem Anwender eine Begründung und Empfehlung für die Korrektur gegeben. Erfolgt diese Korrektur, führt das zur Änderung der Aufgabenstellung. Können die Korrekturen der Funktion aus Gründen der Einordnung in den Systementwurf nicht erfolgen, werden die bisher entstandenen Kosten in Rechnung gestellt.

Bei der Realisierbarkeit des geplanten Gate-Array-Schaltkreises muß der Kunde für *seinen* Schaltkreis ein detailliertes Simulationskonzept für die Verifikation des Entwurfs und ein Prüfkonzept vorgeben (siehe weiter unten). Auf der Basis der Funktionsbeschreibung wird vom Schaltkreiszentrum – wie in /1/ beschrieben – der Gate-Array-Schaltkreis entworfen und simuliert. Auch bei dieser Entwurfsart muß der Anwender nach erfolgreicher Bestätigungssimulation, das heißt nach Erreichen der vorgegebenen Sollparameter, die Erfüllung seiner Aufgabenstellung durch den vorliegenden Entwurf bestätigen. Auch hier wird das Simulationsprotokoll das Freigabedokument. Man erkennt erneut, welche wichtige Rolle die Planung der Simulation und damit die Vorgabe der Simulationseingangsdaten hat. Für alles, was nicht simuliert wurde, hat man keine Garantie, daß es durch den Gate-Array-Schaltkreis realisiert wird.

Nach der Freigabe des Entwurfs übernimmt das Schaltkreiszentrum für beide Entwurfsarten die gleichen Leistungen. So werden die Datenträger erzeugt und an den Hersteller übergeben. Je nach vereinbarter Schnittstelle enthält der Datenträger die Strukturbeschreibung, und der Hersteller generiert Test- und Maskendaten (Bild 13, Variante 4), oder er enthält bereits die Test- und Maskendaten (Variante 5). Wesentlich für die Mittlerrolle des Schaltkreiszentrums ist die Übernahme aller organisatorischen Arbeiten einschließlich Terminbindung und -kontrolle für die Musterpräparation beim Hersteller.

In einem Kundenmeßlabor wird Prüf- und Meßtechnik bereitgestellt, damit kann neben Messungen von elektrischen oder dynamischen Parametern bei Bedarf auch ein Funktionstest des Gate-Array-Schaltkreises durchgeführt werden.

Abschließender Vergleich von Gate-Array-Schaltkreisen mit Standardzellen-Schaltkreisen und Vollkundenwunsch-Schaltkreisen

Im Bild 15 sind die Zeiten für den Entwurf und die Herstellung von Vollkundenwunsch-Schaltkreisen und den beiden ASIC-Varianten qualitativ dargestellt.

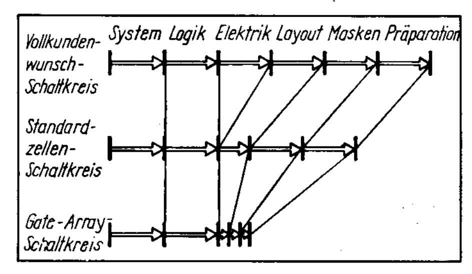


***Bild 15** Vergleich der erforderlichen Zeiten für Entwurf und Herstellung unterschiedlicher Schaltkreisarten*

Es ist ersichtlich, daß neben der Einsparung von Entwurfsaufwand bei Gate-Array-Schaltkreisen auch Herstellungszeit gespart wird, wodurch sich die bei Gate-Array-Schaltkreisen an sich typischen vorteilhaften kurzen Bereitstellungszeiten, insbesondere für die Muster ergeben.

Die Bereitstellungszeiten der ersten Muster schwanken auch international, wobei sie durch die technologisch bedingten Mindestzeiten, durch die Prozeßdisziplin, aber auch durch den planmäßigen Organisationsablauf beeinflußt werden. In jedem Fall sind sie ein charakteristisches Maß für die Leistungsfähigkeit des Herstellers. Für DDR-Anwender können als Grundlage für eine Planung die in /4/ angegebenen Richtwerte angenommen werden:

- Arbeit am CAD-System 3 Monate
- Datenträgererzeugung und -übergabe 1 Monat
- Musterpräparation 4 bzw. 6 Monate.

Das Anstreben kurzer Bereitstellungszeiten ist immens wichtig, damit die Hardwareentwickler den Einsatz von ASICs überhaupt akzeptieren, denn sie müssen ja die gesetzlich fixierten Entwicklungszeiten für die Geräte einhalten. Erst mit den Mustern kann und *muß* die Entwicklung und die Erprobung der Geräte und Systeme, für die die Gate-Array-Schaltkreise entworfen wurden, erfolgen.

Den vielen Vorteilen von Gate-Array-Schaltkreisen stehen natürlich auch Nachteile gegenüber. So weisen Gate-Array-Schaltkreise – prinzipbedingt – eine geringe Transistordichte und daraus resultierend große Chipflächen auf. (Für die zirka 13000 Transistoren beim U 5200-System werden 52 mm^2 benötigt, bei Standardzellen-Schaltkreisen kann die Transistordichte um zirka 1,5 und bei Vollkundenwunsch-Schaltkreisen um zirka 2 größer werden). Daraus folgt, daß Gate-Array-Schaltkreise vorwiegend bei kleinen und mittleren Stückzahlen kostengünstig eingesetzt werden können. Nach internationalen und nationalen Angaben sind das zwischen 500 und 10000 Gate-Array-Schaltkreise eines Typs.

Diese Stückzahlangabe besitzt nur grob orientierenden Charakter. Einerseits gehen bei kleinen Stückzahlen die konstanten Entwurfskosten stärker in den Schaltkreispreis ein, andererseits hängt die ökonomisch vertretbare Stückzahl nur von den erzielten Einsparungen bei Einsatz eines oder mehrerer Gate-Array-Schaltkreise ab. So sinkt natürlich mit einer nachweisbaren großen Materialeinsparung pro Einsatzfall die ökonomische vertretbare Mindeststückzahl und umgekehrt. In /4/ werden als „summarischer Nutzeffekt" bei bereits entwickelten ASICs 50,– bis 400,–M pro Schaltkreis angegeben. Dabei bestimmt der Preis eines verkappten und ausgemessenen Gate-Array-Schaltkreises verständlicherweise beträchtlich die möglichen und notwendigen ökonomischen Effekte und besonders die Einsatzbreite (Konsumgüter!). Er beträgt bei U 5200-Schaltkreisen 200,– bis 250,–M in Abhängigkeit von der Gehäuseform. Jedoch wäre es generell falsch zu fordern, daß der Preis des eingesetzten Gate-Array-Schaltkreises kleiner sein muß als die Preissumme der ersetzten Standard-Schaltkreise. Neben der Materialeinsparung – Schaltkreise, Leiterplattenmaterial, Gehäuse, Netzteil – gilt es, eine Vielzahl anderer Faktoren, die den Preis und die Eigenschaften des Gerätes oder des Systems beeinflussen, zu betrachten, um fundierte Aussagen zum ökonomischen Gesamteffekt machen zu können.

In diesem Zusammenhang soll nochmals erwähnt werden, daß der formale Ersatz vorhandener Schaltungen und ihrer Leiterplatten durch einen oder mehrere Gate-Array-Schaltkreise nur *eine* Variante für die Anwendung ist und nicht die erstrebenswerte. Beim geplanten Einsatz von Gate-Array-Schaltkreisen muß besonders über die möglichen Verbesserungen des Systemkonzeptes zur Erhöhung des Gebrauchswertes, des wissenschaftlich-technischen Niveaus usw. nachgedacht werden, weil durch den Einsatz von ASICs manche zukunftsweisenden Konzepte überhaupt erst möglich werden.

Gate-Array-Schaltkreise sind eine bedeutende Klasse innerhalb der anwendungsspezifischen Schaltkreise. Sie besitzen große Vorteile bei Entwurf und Präparation. Gate-Array-Schaltkreise sind charakterisiert durch

- extrem kurze Entwurfszeiten (0,5–4 Mann-Monate)
- kurze Zeiten für Bereitstellung erster Muster (prinzipiell 4–12 Wochen!)
- kostengünstiger Einsatz bei kleinen bis mittleren Stückzahlen (500–10000 Stück)
- volle Funktionsfähigkeit ohne Redesign.

Damit wird die Artikelfolge zu den Gate-Array-Schaltkreisen abgeschlossen.

Literatur

/1/ Müller, D.: Entwurf von Gate-Array-Schaltkreisen (Teil 1). Mikroprozessortechnik, Berlin 3 (1989) 3, S. 83
/2/ Müller, D.: ASIC – eine Revolution? Mikroprozessortechnik, Berlin 2 (1988) 11, S. 323
/3/ Müller, D.; Fügert, E.: Entwurf von Gate-Array-Schaltkreisen (Teil 2). Mikroprozessortechnik, Berlin 3 (1989) 6, S. 168
/4/ Organisationsprojekt SCHALTKREISENTWURF. Schulungsmaterial des Schaltkreiszentrums des VEB Textima-Elektronik Karl-Marx-Stadt

☒ KONTAKT ☏
Technische Universität Karl-Marx-Stadt, Sektion Informationstechnik, Reichenhainer Straße 70, Karl-Marx-Stadt, 9022; Tel. 5 61 31 95

Turbo-Pascal-Praxis

Teil 4

Manfred Zander, Dresden

Nachdem wir uns im Teil 3 unseres Kurses mit Bildschirmdarstellungen, Hardcopyfunktionen und Disketteninhaltsverzeichnissen beschäftigt haben, wenden wir uns in dieser Folge der Bearbeitung von Dateien anderer Programme zu und werden die Verwaltung von Daten als Listen oder als Bäume beginnen.

Dateien anderer Programme

Bei der Programmierung ist es oft notwendig, eine Datenschnittstelle zu anderen Programmsystemen zu gewährleisten. Der häufigere Fall ist dabei die Übernahme von Ergebnissen aus anderen Programmen. Sind diese Programme nicht in Turbo-Pascal geschrieben, kann dies auf Grund der verschiedenen Dateistrukturen ein Problem werden. Ein oft genutzter Weg ist dann die Übertragung der Daten mit Hilfe von Textdateien. Das bedingt aber immer einen erhöhten Aufwand an Datenkonvertierungen. Entweder müssen im erstellenden Programm diese Text-Dateien für das lesende Programm extra aufgebaut werden, oder aber man muß spezielle Konvertierungsprogramme entwikkeln, um aus binären Dateien nur für die Übertragung Textdateien erstellen zu können. Dabei ist im lesenden Programm dann aber die Rücktransformation in bearbeitbare Daten natürlich auch wieder notwendig.

Typische Fälle sind die Übertragung von dBase II-Datenbankdateien in ein Turbo-Pascal-Programm sowie die Bearbeitung von Bilddateien, die von CAD-Arbeitsplätzen erstellt wurden. In beiden Fällen besitzen die Dateien, die gelesen werden sollen, eine Dateistruktur, die nicht sofort lesbar ist. Da auch bei der File-Arbeit die strenge Typüberwachung im Compiler wirksam ist, gelingt es in den seltensten Anwendungen, diese Klippe durch eine ausgeklügelte Datei-Typisierung zu umschiffen. Es gibt nur zwei Dateitypen, die auf beliebige Dateien angewandt werden können. Dies sind der vordeklarierte Typ **file**, mit dem nur blockweise gearbeitet werden kann und unter einigen Betriebssystemen der Typ **file of char** bzw. **file of byte**.

Prinzipiell ist noch zu sagen, daß vor jeder Erarbeitung einer solchen Lösung eine intensive Analyse der zu lesenden Datendateien notwendig ist. Erst die so gewonnenen Informationen bieten dann die Möglichkeiten zur Lösung der Aufgaben.

dBase II-Dateien

Die Struktur der Datenbankdateien von dBase II ist nicht vollständig durch eine Typvereinigung unter Turbo-Pascal beschreibbar. Diese Tatsache resultiert vor allem daraus, daß die Dateien aus einem Dateikopf bestehen, der völlig anders strukturiert ist als die folgenden Teile der Datei, in der die Daten gespeichert sind. Die Struktur des Dateikopfes ist bei allen Datenbankdateien dieses Programmsystems gleich und kann unter Turbo-Pascal durch einen kombinierten Record- und Array-Typ beschrieben werden. In diesem Kopf einer konkreten Datenbankdatei stehen dann die Informationen darüber, aus wievielen und welchen Feldern die folgenden Datensätze aufgebaut sind. Auf die Angabe einer konkreten Realisierung zum Lesen und Schreiben von DBF-Dateien kann hier verzichtet werden, da es bereits eine ausreichende Anzahl von Veröffentlichungen zu diesem Thema gibt.

CAD-Bilddateien

In den letzten Jahren hat die Verbreitung von CAD-Grafikarbeitsplätzen rapide zugenommen. Diese neuen Arbeitsplätze erlauben es, Konstruktionsarbeiten am Grafikbildschirm mit bedeutend verringertem Zeitaufwand qualitativ hochwertig zu erledigen. Das primäre Ergebnis einer so durchgeführten Konstruktion ist aber immer eine Datei, in der die erarbeitete Grafik gespeichert ist. Was nun fehlt, ist ein Bindeglied zwischen dieser Datei und der jeweils vorhandenen Technik und Technologie. Ein sofortiger Austausch dieser Technik ist in den seltensten Fällen ökonomisch sinnvoll oder möglich. Es besteht also das Bedürfnis, die auf den Grafikarbeitsplätzen entstehenden Bilddateien maschinell lesen zu können, um eine Umwandlung der enthaltenen Informationen in die vorgeschriebenen Datenformate erreichen zu können. Man kann zwar die Konstruktionsergebnisse der neuen Systeme plotten und diese geplotteten Bilder dann wieder digitalisieren, aber zum Standard sollte dieses Vorgehen nicht erhoben werden. Es muß eine Möglichkeit der maschinellen Datenübertragung zwischen den verschiedenen Systemen gefunden werden. Diese Übertragung sollte in beide Richtungen vorgesehen werden. Daher ergibt sich also auch die Aufgabe, die Bilddateien der neuen Systeme schreiben zu können. Solche Lösungen können selbstverständlich auch dazu genutzt werden, Funktionen zu entwickeln, die das CAD-System selbst nicht bietet. Dabei wird dann eine Bilddatei des Systems gelesen und der bearbeitete Stand dieser Datei im gleichen Format wieder auf Diskette oder Festplatte abgelegt.

Als Beispiel wird im folgenden der binäre Datenaufbau des sehr weit verbreiteten CADdy 3.04 erläutert und eine Lösung für das Lesen und Schreiben der Bilddateien, dieses auf 16-Bit-Technik laufenden CAD-Systems geboten. Wie das genannte CAD-System selbst, erfordert auch die vorgestellte Lösung einen 16-Bit-Rechner und MS-DOS. Weiterhin muß diese Lösung mit dem speziellen Compiler Turbo-87 übersetzt werden. Dieser Compiler, der den Koprozessor 8087 unterstützt, gestattet die Berechnung der hochgenauen Realzahlen, die in der Lösung benötigt werden.

Die Struktur der Bilddateien des genannten Systems kann auf den ersten Blick betrachtet mit einer variablen Typvereinbarung (einer Datensatz-Variante) beschrieben werden. Jeder einzelne Grafiksatz, der beispielsweise eine Linie oder einen Punkt beschreibt, hat an seinem Anfang eine konstante Struktur. In dieser Kopfstruktur existiert jeweils ein Zeichen zur Kennzeichnung des folgenden Inhalts. In Abhängigkeit von diesem Zeichen enthält der Rest des Datensatzes alle notwendigen Informationen, um einen Punkt oder eine Linie zu beschreiben. Natürlich sind diese Informationen für verschiedene grafische Elemente auch verschieden aufgebaut und belegen jeweils eine unterschiedliche Anzahl Bytes. Leider ist es aber trotzdem nicht möglich, einen solchen kompakten Satztyp zu erstellen und die Bilddateien mit ihm zu typisieren. Dies hat vor allem zwei Gründe. Als erstes ist das Entscheidungszeichen im konstanten Satzkopf nicht die letzte Angabe im Kopf. Diese Klippe kann zwar umschifft werden, ein zweiter Grund macht dieses einfache Vorgehen aber unmöglich: Die einzelnen Datensätze sind in der Datei mit ihren verschiedenen Größen nahtlos aneinandergefügt. Damit ist eine Grundeigenschaft typisierter Files, die konstante Satzlänge, nicht mehr gegeben. Es bleibt also noch der Weg, die Bilddateien als *file of byte* zu behandeln.

Das Prinzip der im Bild 36 vorgestellten Lösung ist im Verhältnis zu ihrem Schreibaufwand einfach. Die Prozedur *Satzlesen* gestattet das Einlesen eines kompletten Datensatzes. Dazu liest sie zuerst byteweise den konstant typisierten Kopf eines Satzes in das Array *Kopfbuffer*. Dieses Feld ist der Variablen *Kopf*, die die wahre Struktur dieses Satzkopfes enthält, überlagert. Hier sind konstant ein Folgenkennzeichen des Grafikelementes, dessen Farbe und dessen Zeichenebene beschrieben. Für die weitere Organisation sind die folgenden Variablen *Was* und *Zahl* im Kopf von Bedeutung. *Was* gibt an, welche Art von Bildelement folgt, und in *Zahl* ist die Länge des restlichen Datensatzes gespeichert. Wenn in *Was* ein **V** steht, das heißt das Versionskennzeichen folgt, wird dieses abgetestet. Handelt es sich in *Was* um eine andere Kennung, wird der Rest des Satzes lediglich auf das Array *Buffer* aufgelesen. Um die Anzahl der hierbei zu lesenden Bytes zu ermitteln, sind von der Variablen *Kopf.zahl* lediglich 16 Byte zu substrahieren.

Dem Array *Buffer* sind alle möglichen Reststrukturen eines Grafik-Satzes überlagert. Das lesende Programm kann sich nun anhand der Variablen *Kopf.Was* informieren, welche Art von Bildelement eingelesen wurde und daraufhin die entsprechenden Informationen aus dem zugehörigen überlagerten Feld entnehmen. Beim nächsten Aufruf der Prozedur Satzlesen werden diese Daten durch den neuen Satz überschrieben. Die Tafel 1 gibt an, welche Kennungen für welche grafischen Elemente im Feld *Kopf.Was* genutzt werden. Die Prozedur Satzschreiben realisiert die Umkehrung des beschriebenen Vorganges. Sie speichert den grafischen Sachverhalt, der zuvor in den entsprechenden Feldern abgelegt sein muß, in eine Bilddatei. Dabei orientiert sie sich wieder an der

```
type Picfiletype = file of char;
var Kopf : record                          (* Feste Kopf-Struktur  *)
             folge,farbe,ebene:integer;     (* aller Graphik-Saetze *)
             Was:char; zahl :byte
           end;
    Kopfbuffer : array[1..8] of char absolute Kopf;
    Buffer:array[1..128] of char;
    Apuf : record
             y,x,w,zy,zx:real;name:string[8]
           end absolute Buffer;
    aapuf: record
             y,x:integer;Name:string[7]
           end absolute Buffer;
    Cpuf : record
             y,x,th:real; Ctb,Cta,Ctw:integer;
             ttext:string[15]; INT1,int2:real
           end absolute Buffer;
    ddpuf: record
             y,x,th:real;tb,ta,tw:integer;
             ttext:string[15]
           end absolute Buffer;
    Dpuf : record
             y,x,th:real;tb,ta,tw:integer;
             ttext:string[79]
           end absolute Buffer;
    Epuff :char absolute Buffer;
    Kpuf : record
             y,x,w2,w1:real;ls,lTyp:integer;r:real;
             lb,la:integer
           end absolute Buffer;
    Lpuf : record
             y1,x1,y2,x2:real; lTyp,ls:integer
           end absolute Buffer;
    mmpuf: record
             y1,x1,y2,x2,y3,x3:real
           end absolute Buffer;
    Mpuf : record
             y1,x1,y2,x2,y3,x3:real
           end absolute Buffer;
    Npuf : record
             y1,x1,y2,x2:real
           end absolute Buffer;
    Opuf : record
             y,x:real
           end absolute Buffer;
    Qpuf : record
             bgy,bgx:real
           end absolute Buffer;
    Rpuf : record
             zoomY,zoomX:real
           end absolute Buffer;
    Vpuf : record
             c1,c2,c3,c4:char; i1,i2,i3:integer
           end absolute Buffer;
    Xpuf : record
             Refy,Refx:real
           end absolute Buffer;

procedure Satzlesen(var datei:Picfiletype);
  var i:integer;
  begin
   for i:=1 to 8 do read(datei,Kopfbuffer[i]);
   if Kopf.Was = 'V'
     then begin for i:=1 to 10 do read(datei,Buffer[i]);
            if (Vpuf.c1='3')and(Vpuf.c2='.') and
               (Vpuf.c3='0')and(Vpuf.c4='4') then write(' 3.04 ')
              else writeln('Datei hat nicht das Format von CADdy 3.04 ');
          end
     else for i:=1 to Kopf.zahl-16 do read(datei,Buffer[i]);
  end;

procedure Satzschreiben(var datei:Picfiletype);
  var i:integer;
  begin
  with Kopf do
   case Was of
    'a':      zahl:=$1C;        'A':      zahl:=$41;
    'E':      zahl:=$10;        'K':      zahl:=$3C;
    'L':      zahl:=$34;        'M','m':  zahl:=$40;
    'N':      zahl:=$30;        'V':      zahl:=$4;
    'd','D':  zahl:=$1F+$10+length(ddpuf.ttext);
    'C':      zahl:=$2F+$10+length(Cpuf.ttext);
    'X','R','O','Q':zahl:=$20;
   end;
  for i:=1 to 8 do write(datei,Kopfbuffer[i]);
  if Kopf.Was = 'V'
    then begin
           Vpuf.c1:='3';Vpuf.c2:='.';Vpuf.c3:='0';Vpuf.c4:='4';
           Vpuf.i1:=0;Vpuf.i2:=0;Vpuf.i3:=0;
           for i:=1 to 10 do write(datei,Buffer[i]);
         end
    else for i:=1 to Kopf.zahl-16 do write(datei,Buffer[i])
  end;

Procedure write_L(var Datei:PicFileType;x,y,xx,yy:real;e,col:integer);
  begin with Lpuf do begin
    x1:=x;x2:=xx;y1:=y;y2:=yy;lTyp:=1;ls:=0 end;
    with kopf do begin ebene:=e;farbe:=col;folge:=0;Was:='L' end;
    satzschreiben(Datei)
  end;

Procedure write_Q(var Datei:PicFileType;x,y:real);
  begin with Qpuf do begin bgx:=x;bgy:=y;end;
    with kopf do begin ebene:=0;farbe:=0;folge:=0;Was:='Q' end;
    satzschreiben(Datei)
  end;

Procedure write_V(var Datei:PicFileType);
  begin
    with kopf do begin ebene:=0;farbe:=0;folge:=0;Was:='V' end;
    satzschreiben(Datei)
  end;

Procedure write_E(var Datei:PicFileType);
  begin
    with kopf do begin ebene:=0;farbe:=0;folge:=0;Was:='E' end;
    satzschreiben(Datei)
  end;
```

Bild 36

Tafel 1 Übersicht

Kennung	Daten in	Element
A	Apuf	Symbol
a	aapuf	Kurz-Symbol
C	Cpuf	Bemaßungstext
d	ddpuf	kurzer Text
D	Dpuf	langer Text
E	–––	Ende-Kennung
K	Kpuf	Kreis
L	Lpuf	Linie
m	mmpuf	Dreieckiger Füllblock
M	Mpuf	Parallelogram Füllblock
N	Npuf	Rechteckiger Füllblock
O	Opuf	Punkt
Q	Qpuf	Bildmaße
R	Rpuf	Faktor für Kurzsymbole
V	Vpuf	Versionskennzeichen
X	Xpuf	Symbol-Referenzpunkt

Variablen Kopf.Was, die das rufende Programm einstellen muß. Auf der Grundlage dieser Information ermittelt sie die Länge des zu schreibenden Satzes. Das Schreiben selbst wird wiederum in zwei Etappen durchgeführt. Zuerst wird der Satz-Kopf abgespeichert und dann der entsprechende Rest des Satzes. Lediglich das Versionskennzeichen erfährt eine Sonderbehandlung.

Mit Hilfe dieser beiden Prozeduren ist es nun möglich, Bilddateien sowohl zu lesen als auch zu schreiben. Im Gegensatz zum Lesen, bei dem es der jeweiligen Anwendung überlassen bleibt, welche gelesenen Informationen weiter verwendet werden, muß beim Schreiben stets garantiert werden, daß die abzuspeichernden Datensätze komplett gefüllt sind. Dabei genügt es nicht, der Variablen *Lpuff.1Typ*, also dem Linientyp einmalig einen Wert zuzuweisen. Bei jeder Schreib- oder Leseoperation eines anderen grafischen Elementes kann dieser Wert im Buffer überschrieben werden, da ja alle Buffer-Variablen überlagert sind. Daher ist es sinnvoll, diese Aufgabe speziellen Prozeduren anzuvertrauen, die jeweils einen einzigen Grafik-Satz zum Schreiben anweisen. In diesen Prozeduren können dann auch, wie in den folgenden Beispielen, Standardbelegungen fest programmiert werden.

Als Beispiel soll hier lediglich die Prozedur write_L erläutert werden. Sie schreibt eine Linie in die Bilddatei. Der Prozedur werden als Parameter Anfangs- und Endpunkte der zu speichernden Linie, deren Zeichenebene und Farbe übergeben. Diese Angaben werden korrekt in den Übertragungspfuffern gesetzt. Dabei wird der Linientyp, die Linienstärke und das Folgen-Kennzeichen mit konstanten Werten eingestellt. Wesentlich ist wieder das Setzen des Satzkennzeichens *Kopf.Was*. Somit sind alle zu übertragenden Informationen bereitgestellt und die Prozedur Satzschreiben kann aufgerufen werden. Erst sie schreibt ja die Daten in die Bilddatei.

Zum Schluß sei noch darauf hingewiesen, daß in allen Prozedur-Vereinbarungen enthaltene File-Variable *Datei* natürlich im

```
Program Linien_Selektieren;

{$I PIC.BIB }

var cad,cadneu:PicFileType:

begin
  assign(cad,'Bild.pic');reset(cad);
  assign(cadneu,'Linien.pic');rewrite(cadneu);
  repeat
    satzlesen(cad);
    if kopf.was in ['V','Q','L','E'] then
      satzschreiben(cadneu)
  until Kopf.was = 'E';
  close(cad);
  close(cadneu)
end.
```

Bild 37

Hauptprogramm erst zugewiesen werden muß. Weiterhin ist es nicht möglich, auf eine Bilddatei gleichzeitig schreibend und lesend zuzugreifen.

Zur Demonstration der Nutzung dieser Lösung geben wir im Bild 37 ein kleines Programmbeispiel an, welches eine Bilddatei liest und lediglich die Linien dieser Datei auf eine zweite Datei überträgt. Die Sätze mit den Kennungen V, Q oder E, die zu jeder Bilddatei gehören, werden natürlich ebenfalls übernommen. Die in der Include-Anweisung im Programmtext angegebene Datei PIC.BIB enthält die im Bild 36 abgedruckte Lösung.

Listen

Nachdem wir in den letzten Kapiteln darüber gesprochen haben, wie wir uns Daten beschaffen, wollen wir nun die Verwaltung von Daten im Hauptspeicher betrachten. Es wird dabei vor allem um die effektive Haltung größerer Datenmengen gehen. Die hier vorzustellende Methodik der Daten-Listen ist hauptsächlich für jene Fälle gedacht, in denen der Programmierer beim Entwurf eines Programms nicht weiß, welche konkreten Datenmengen während eines Programmlaufes auftreten werden. Zum Verständnis der folgenden Lösungen ist die Vertrautheit des Lesers mit dem Pointer-Begriff und der Handhabung von dynamischen Variablen Voraussetzung. Diese Beispiele beschäftigen sich durchweg mit Listen- und Baum-Strukturen, zu deren Beschreibung es in der Fachliteratur im allgemeinen feste Sprachregelungen gibt. Der Autor wird sich aber nur lose an diesen Sprachgebrauch halten, da er eine größtmögliche Verständlichkeit speziell für Turbo-Programmierer erreichen möchte.

Einfach gekoppelte, geordnete Listen

Die in der ersten Lösung zu besprechenden Listen sollen einfach gekoppelt und geordnet sein. Sie besitzen also genau ein erstes und ein letztes Listenelement. Jedes Listenelement enthält neben dem *Eintrag* eine Variable mit Namen *Naechster*, in dem auf das jeweils nächste Element der Liste verwiesen wird. Dieser Verweis ist ein Pointer. Das letzte Listenelement enthält in diesem Feld den Pointerwert *NIL*. NIL ist ein Pointer der

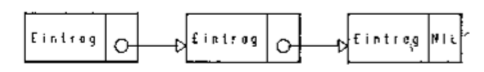


Bild 38

nirgendwohin zeigt und hier als Listen-Endekennung verwendet wird. Man kann sich also mit Hilfe der in *Naechster* enthaltenen Pointerwerte innerhalb dieser Liste vorwärts tasten, bis deren Ende erreicht ist. Ein Rückwärtslaufen in der Liste ist dagegen hier nicht möglich, da in den Listenelementen kein Verweis auf den Vorgänger enthalten ist, was natürlich auch möglich wäre. Jedes Listenelement enthält also eine Kopplung zu dem Nachfolgeelement. Dagegen enthält ein solches Element keinen Hinweis auf seinen Vorgänger. Damit ist die Liste nur einfach gekoppelt (siehe Darstellung in Bild 38).

Zum Bearbeiten solcher Listen dienen die drei in Bild 39 gezeigten Funktionen. Das Einfügen eines neuen Listenelementes, das Suchen und das Löschen eines Elements. Dabei soll beim Einfügen für das neue Element Platz im Hauptspeicher reserviert werden und beim Löschen soll dieser Platz wieder freigegeben werden. So wird erreicht, daß die Liste jeweils nur den Platz im Hauptspeicher beansprucht, den sie aktuell benötigt.

Wenn die Elemente der Liste geordnet gespeichert werden sollen, ist ein Ordnungskriterium notwendig. Da diese Ordnungsvorschrift aber für verschiedene Listeninhalte sehr unterschiedlich sein kann, wird für die vorgestellten Listenoperationen die Funktion *kleiner* vorausgesetzt. Selbstverständlich muß auch die Definition des Typs *Listeninhalt* außerhalb dieser Lösung erfolgen, da dieser ja für jede Anwendung unterschiedlich sein wird (Bild 39).

Die erste Komponente der Lösung ist die Prozedur Einfügen. Ihr werden zwei Parameter übergeben. Der erste, mit L bezeichnete, ist ein Pointer, der auf den Anfang der zu erweiternden Liste zeigt. Der zweite Parameter, mit E bezeichnet, ist der Eintrag, der in die Liste aufgenommen werden soll. Zur internen Organisation dieser Prozedur sind zwei weitere Pointervariablen p und q notwendig. Der Pointer p wird zuerst auf den angegebenen Anfang der Liste eingestellt. Die folgende Suchschleife wird genau dann verlassen, wenn die Stelle in der Liste gefunden wurde, an der der Eintrag eingefügt werden soll. Ist die angegebene Liste leer, enthielt der Parameter L also den Wert NIL, erfolgt kein Suchen. Nach dieser Schleife wird der notwendige Speicherplatz für das neue Listenelement mit der Standardprozedur *new* reserviert. Der Pointer p zeigt jetzt auf das neue Element, und es wird mit dem Inhalt des Eintrags E gefüllt. Nun muß dieses erzeugte Listenelement auch noch logisch in die Liste eingebunden werden, das heißt, die Felder *Naechster* des Vorgängers und natürlich auch des neuen Elementes selbst müssen korrekt auf den jeweiligen Nachfolger eingestellt werden. Erst so wird das Element in die Listenordnung eingefügt. Dazu ist aber eine Fallunterscheidung notwendig. Hat der Pointer q nach dem Suchvorgang immer noch den Wert NIL, so muß das Element am Anfang der Liste eingefügt werden, oder aber die Liste existierte noch gar nicht. In diesem Fall wird dem Feld *Naechster* des neuen Elementes der alte Listenanfang (L) übergeben und die Procedur gibt den neuen Listenanfang zurück. Im anderen Fall erfolgt ein echtes Einfügen in die alte Liste. Dazu muß der Vorgänger, auf den jetzt die Variable q zeigt, auf das neue Element zeigen, und dieses muß auf das folgende Element zeigen, auf das zuvor der Vorgänger zeigte. Eine Sonderbetrachtung für das Listen-Ende ist dabei nicht notwendig.

Der Prozedur Löschen werden zwei Pointer als Parameter übergeben. Der erste Pointer (L), ist wiederum ein Zeiger auf den Anfang der zu behandelnden Liste. Der zweite Parameter, der Pointer P, zeigt auf das zu löschende Element der Liste. Dabei muß gesagt werden, daß diese Prozedur nicht prüft, ob P wirklich auf ein Listenelement zeigt. Ist dies nicht der Fall, kann das Schaden anrichten. Zur internen Arbeit der Prozedur ist eine weitere Pointervariable (q) notwendig. In der Prozedur müssen drei Fälle unterschieden werden. Im ersten Fall zeigt das zu löschende Element noch auf ein weiteres Listenelement. Dann ist das Löschen auf eine sehr einfache Art und Weise möglich. Das zu

```
(* G_Liste.INC......Operationen zu geordneten Listen *)

type Liste          = ^ListenEintrag;
     ListenEintrag = record
                       Eintrag     : Listeninhalt;
                       Naechster   : Liste
                     end;
procedure Einfuegen (var L:Liste;E:ListenInhalt);
  var p,q:Liste;
  begin
    q:=NIL;p:=L;
    while p<>NIL do with p^ do
      if kleiner(Eintrag,E)
        then begin q:=p;p:=naechster end
        else p:=NIL;
    new(p);p^.Eintrag:=E;
    if q=NIL
      then begin p^.Naechster:=L;L:=p end
      else begin p^.Naechster:=q^.Naechster;q^.Naechster:=p end;
  end;

procedure Loeschen (var L:Liste;P:Liste);
  var q:Liste;
  begin
    q:=P^.Naechster;
    if q<>NIL
      then begin P^:=q^;dispose(q) end
      else
        if P=L
          then begin L:=NIL;dispose(p) end
          else
            begin q:=L;
              while q^.Naechster<>P do q:=q^.Naechster;
              q^.Naechster:=P^.Naechster;dispose(P)
            end
  end;

procedure Suchen(L:Liste;E:ListenInhalt;var P:Liste);
  var gefunden:boolean;
  begin
    P:=L;gefunden:=false;
    while ( P<>NIL ) and ( not gefunden ) do
      begin
        gefunden:= not( kleiner(P^.Eintrag,E) or kleiner(E,P^.Eintrag) );
        if not gefunden then P:=P^.Naechster
      end;
  end;
```

Bild 39

löschende Listenelement wird mit seinem Nachfolger überschrieben, und der alte Platz des Nachfolgers wird freigegeben. Jetzt zeigt der Vorgänger des gelöschten Elements auf dessen Nachfolger, da dieser ja den Platz des gelöschten Elementes eingenommen hat, und dieser zeigt weiterhin auf dessen Nachfolger. Die logische Folge der Elemente wurde also erhalten. Der zweite Fall, der eintreten kann, ist gegeben, wenn das zu löschende Element einer Liste das einzige in dieser Liste ist. Tritt dies ein, wird dem rufenden Programm über die Variable L mitgeteilt, daß die Liste nun nicht mehr existiert, und der Platz des letzten Elementes dieser Liste wird freigegeben. Am schwierigsten gestaltet sich der Löschvorgang, wenn das zu löschende Element das letzte einer Liste ist, die jedoch später noch weitere Elemente aufnehmen soll. Hierbei ist es notwendig, den Vorgänger des zu löschenden Elementes zu ermitteln, da ja in ihm der Verweis auf das nächste Element gelöscht werden muß. Die Ermittlung des Vorgängers ist bei der gewählten Listenorganisation nur möglich, indem wir uns vom Anfang der Liste bis an die interessierende Stelle vortasten. Nachdem im so gefundenen Vorgänger der Verweis auf das Folgeelement mit NIL überschrieben wurde, kann der belegte Platz des gelöschten Elementes freigegeben werden.

Die dritte Listenoperation wurde zum Suchen eines Listenelementes entwickelt. Ihr wird mit Hilfe des Parameters L wiederum ein Zeiger auf den Anfang der zu durchsuchenden Liste übergeben. Der Parameter E enthält den Eintrag, der gesucht werden soll, und über den Parameter P gibt sie den Zeigerwert zurück, der auf das gefundene Listenelement zeigt. Enthält die Liste das zu suchende Element nicht, wird über diesen Parameter der Wert NIL zurückgegeben. Die Prozedur enthält im wesentlichen nur die Suchschleife selbst. Der in ihr enthaltene Test sieht auf den ersten Blick etwas kompliziert aus. Er nutzt die vorausgesetzte Funktion *kleiner* und prüft die Identität.

Zur Illustration zeigt Ihnen Bild 40 ein kleines Programm. In ihm werden als Listenelemente einfache Integer-Zahlen vereinbart, und für diese Zahlen wird auch die Boolesche Funktion kleiner aufgestellt. Zuerst wird der Pointer-Variablen L, die jeweils den Listenanfang markieren soll, der Wert NIL zugewiesen. Danach wird die Liste mit 20 zufälligen Werten gefüllt. Diese Werte werden geordnet in der Liste abgespeichert. Davon überzeugt uns die dann folgende Schleife, die den entstandenen Listeninhalt anzeigt. Nun werden alle Zahlen mit dem Wert 1 aus der Liste entfernt und der verbleibende Rest der Liste wird wiederum ausgegeben. Bei allen diesen Aktionen dient der Pointer P als Zeiger zu Listenelementen, die jeweils adressiert werden sollen. Die erste und letzte Anweisung im Programm garantieren, daß nach dem Programmlauf der Speicherplatz aller dynamischen Variablen wieder freigegeben ist.

Zusammenfassend: Der für die Verwaltung der Liste erforderliche Speicherplatz ist mit einem Pointer in jedem Listenelement auf ein Minimum beschränkt. Dies hat zur Folge, daß die aufgebauten Listen sofort nur vorwärts, nicht aber auch rückwärts durchgemustert werden können. Anders gesagt: Es ist nicht trivial, zu einem Element den Vorgänger zu ermitteln. Weiterhin sind sowohl in der Komponente Einfügen wie auch in der Komponente Löschen Schleifen enthalten, die natürlich Zeit benötigen. Will man die Elemente der Liste wirklich geordnet haben, ist dies nicht zu ändern. Handelt es sich aber um Listeninhalte, die gar nicht geordnet benötigt werden, sollte man die folgende Lösung benutzen.

Doppelt gekoppelte, ungeordnete Listen

Als eine zweite Variante hier nun eine Lösung für die Verwaltung von doppelt gekoppelten, ungeordneten Listen. Damit wird das Suchen sowohl vorwärts als auch rückwärts möglich sein. Dies erfordert aber, in jedem Listenelement zwei Verweise auf die Nachbarelemente in der Liste zu führen. Ein Verweis zeigt dabei auf das jeweils vorhergehende Element, und der zweite Verweis zeigt auf das nachfolgende Element. Die Liste ist somit *doppelt gekoppelt.* Im zweiten Element einer Liste sind also folgende Verweise enthalten: *Vorgaenger* zeigt auf das erste Element und *Nachfolger* zeigt auf das dritte Element der Liste. Das erste Element einer Liste enthält daher im Feld Vorgaenger NIL und das letzte Element einer Liste enthält im Feld Nachfolger NIL (Bild 41). Somit ist ein Durchlaufen der Liste in beiden Richtungen leicht möglich. Da auch auf ein Ordnungskriterium für die Listeninhalte verzichtet wurde, sind die Prozeduren zur Arbeit mit der Liste sehr einfach. Auf Schleifen konnte außer beim Suchen selbst verzichtet werden (Bild 42).

Die Prozedur *Dazu* ermöglicht das Einfügen eines neuen Elementes mit dem Eintrag E in die Liste, auf deren erstes Element der Pointer L zeigt. Da wir ja keine Ordnung in der Liste benötigen, kann das Einfügen des neuen Elementes an beliebiger Stelle der Liste erfolgen. Am einfachsten ist das Einfügen am Anfang der Liste, wie Sie es hier sehen. Zuerst wird für die dynamische Variable, die das einzufügende Element aufnehmen wird, Platz im Hauptspeicher reserviert und sie dann gefüllt. Da es das neue erste Element der Liste sein wird, wird dem Feld Vorgänger der Wert NIL zugewiesen. Das Feld Nachfolger dagegen zeigt auf das alte erste Element der Liste, wobei es unerheblich ist, ob diese Liste bereits existierte. Das alte erste Element der Liste muß nun natürlich in seinem Feld Vorgänger auf das neue erste Element der Liste verweisen. Diese Zuweisung erfolgt nur, wenn die Liste zuvor bereits Elemente enthielt. Die Prozedur Dazu übergibt dem rufenden Programm über den Parameter L den Zeiger auf den neuen Anfang der Liste.

wird fortgesetzt

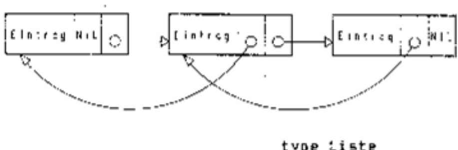


Bild 41

```
Program Integer_Sortieren:

type Listeninhalt = integer;

function kleiner(x,y:integer):boolean;
  begin
    if x<y then kleiner:=true else kleiner:=false
  end;

(*I G_Liste.INC )

var i:integer;
    Heaptop,L,P:Liste;
begin
  Mark(heaptop);
  L:=NIL;
  for i:=1 to 20 do Einfuegen(L,random(10));
  P:=L;
  while P<>NIL do
    begin write(P^.Eintrag:4); P:=P^.Naechster end;
  Suchen(L,1,P);
  while P<>NIL do
    begin Loeschen(L,P); Suchen(L,1,P) end;
  P:=L;
  while P<>NIL do
    begin write(P^.Eintrag:4); P:=P^.Naechster end;
  release(heaptop);
end.
```

Bild 40

```
type Liste          = ^ListenEintrag;
     ListenEintrag = record
                       Eintrag    : Listeninhalt;
                       Vorgaenger,
                       Nachfolger  : Liste
                     end;

procedure Dazu (var L:Liste;E:ListenInhalt);
  var p:Liste;
  begin
    new(p);p^.Eintrag:=E;p^.Vorgaenger:=NIL;p^.Nachfolger:=L;
    if L<>NIL then L^.Vorgaenger:=p;
    L:=p;
  end;

procedure Weg (var L:Liste;P:Liste);
  var V,N:Liste;
  begin
    N:=P^.Nachfolger;V:=P^.Vorgaenger;
    if N<>NIL then N^.Vorgaenger:=V;
    if V<>NIL then V^.Nachfolger:=N else L:=N;
    dispose(P);
  end;

procedure Finde(L:Liste;E:ListenInhalt;var P:Liste);
  begin
    P:=L;
    while (P<>NIL) and (not(P^.Eintrag = E)) do P:=P^.Nachfolger;
  end;

Procedure Ende(L:Liste;var P:Liste);
  begin
    P:=L;
    while P^.Nachfolger<>NIL do P:=P^.Nachfolger;
  end;
```

Bild 42

Überlagerung von Bildschirmfenstern

Ralf Liske
VEB Kombinat ILKA Luft- und Kältetechnik, Stammbetrieb Forschung und Technik Dresden

Der sinnvolle Einsatz der Fenstertechnik in der Dialoggestaltung kann die Benutzerfreundlichkeit von Softwareprodukten insbesondere für den mit dem Computer wenig vertrauten Nutzer wesentlich erhöhen. Die entsprechenden Programmierwerkzeuge erscheinen jedoch für kleine Anwendungsprogramme als zu gewaltig, so daß der Wunsch nach weniger komplexen Hilfsmitteln besteht. Der vorliegende Beitrag zeigt, wie in Turbo-Pascal Version 3.xx unter dem Betriebssystem MS-DOS 3.20 mit geringem Aufwand sich überlagernde Bildschirmfenster programmiert werden können. Damit stehen vielfältige Möglichkeiten zu eleganter Dialoggestaltung offen.

Zwei Prozeduren gestatten das Sichern und Rückspeichern von beliebigen Bildschirmbereichen. Sie erfordern einen Color Graphics Adapter (CGA-Karte) oder einen Monochrombildschirmadapter (bzw. einen dazu kompatiblen, z. B. Hercules-Karte) und laufen auf dem A 7150.
Bild 1 zeigt die beiden Prozeduren in Verbindung mit globalen Deklarationen:

SAVE_WINDOW sichert Bildschirmausschnitt in einem dynamisch verwalteten Speicherbereich,
RESTORE_WINDOW dient zum Rückspeichern des jeweils letzten Bildschirmausschnitts.

Die zu sichernden Bildschirmausschnitte werden mit Hilfe der Pointervariablen *screen_pointer* im Heap abgespeichert. Dazu wird der Bildwiederholspeicher zeilenweise ausgelesen, wobei seine Basisadresse vom Bildschirmadapter abhängt und mit der Funktion 15 des BIOS-Interrupts 16 bestimmt werden kann (siehe auch /1/). Der Bildwiederholspeicher selbst setzt sich zeilenweise aus 25 mal 80 (bzw. 40) Bytepaaren zusammen, wobei das erste Byte den Code des darzustellenden Zeichens und das zweite Byte die Darstellungsattribute (invers, blinkend, Farbe, usw.) enthält. Entsprechend berechnen sich die Adressen der zu sichernden Bildschirm-

```
{ Turbo-Pascal-Include-Datei: WDW-IBM.INC, 23.10.1988 }

CONST bss=80; bez=25;
TYPE  screen_typ          =ARRAY[1..4000] OF byte;
      screen_pointertyp =^screen_pointer_rec;
      screen_pointer_rec=RECORD
                          pre     :screen_pointertyp;
                          x1,y1,
                          x2,y2,zl:byte;
                          scr     :screen_typ
                         END;
      registertyp       =RECORD
                          ax,bx,cx,dx,bp,
                          si,di,ds,es,flags:integer
                         END;
VAR   screen_pointer    :screen_pointertyp;
      basis_screen      :integer;
      register          :registertyp;

{************************************************************}

PROCEDURE SAVE_WINDOW(x1,y1,          { linke obere Ecke }
                      x2,y2:byte); { rechte untere Ecke }
VAR  help            :screen_pointertyp;
     dx,dy,i,x0,xmax:byte;
      j              :integer;
BEGIN

   { Bildschirmmodus bestimmen }
   register.ax:=$0F00; { Lesen aktueller Status }
   intr(16,register);
   CASE register.ax AND $FF OF
     0,1 :BEGIN basis_screen:=$B800; xmax:=bss SHR 1 END;
                                                  { CGA40 }
     2,3 :BEGIN basis_screen:=$B800; xmax:=bss END; { CGA80 }
     7   :BEGIN basis_screen:=$B000; xmax:=bss END;  { Mono }
     ELSE BEGIN                                 { unbekannt }
           clrscr;
           write('FEHLER in SAVE_WINDOW: ');
           writeln('falscher Videomodus !');
           halt
         END
   END; { case }

   { Test, ob gueltiger Bildschirmbereich }
   IF x1<1 THEN x1:=1;       IF x2>xmax THEN x2:=xmax;
   IF x2<x1 THEN x2:=xmax;
   IF y1<1 THEN y1:=1;       IF y2>bsz THEN y2:=bsz;
   IF y2<y1 THEN y2:=bsz;

   { Fensterdimension bestimmen }
   dx:=2*succ(x2-x1);  dy:=succ(y2-y1);

   { Speicherplatz reservieren }
   j:=sizeof(help^.pre)+5*sizeof(help^.x1)+dx*dy;
   IF (j SHR 4 + 1)>maxavail THEN
   BEGIN
     clrscr;
     writeln('FEHLER in SAVE_WINDOW: Heap ist voll !');
     halt
   END;
   getmem(help,j);

   { Fensterdimension Hilfspointer eintragen }
   help^.x1:=x1;  help^.y1:=y1; help^.x2:=x2;  help^.y2:=y2;
   help^.zl:=xmax;

   { Fenster in Hilfspointer kopieren }
   x0:=pred(x1) SHL 1;
   FOR i:=y1 TO y2 DO
     move(mem[basis_screen:(pred(i)*xmax SHL 1 + x0)],
          help^.scr[(i-y1)*dx+1],dx);

   { Verweisstruktur aufbauen }
   IF screen_pointer<>NIL THEN help^.pre:=screen_pointer
                           ELSE help^.pre:=NIL;
   screen_pointer:=help

END; { SAVE_WINDOW }

{************************************************************}

PROCEDURE RESTORE_WINDOW(flag:boolean);
VAR  help       :screen_pointertyp;
     dx,dy,i,x0:byte;
BEGIN

   { Test, ob leere Pointerliste }
   IF (screen_pointer=NIL) THEN
   BEGIN
     clrscr;
     writeln('FEHLER in RESTORE_WINDOW: Pointerunterlauf !');
     halt
   END;

   { Hilfszeiger belegen, Fensterdimension bestimmen }
   help:=screen_pointer;
   dx:=succ(help^.x2-help^.x1) SHL 1;
   dy:=succ(help^.y2-help^.y1);

   { Fenster von Hilfspointer zum Bildwiederholspeicher }
   { kopieren, wenn flag=true }
   x0:=pred(help^.x1) SHL 1;
   IF flag THEN
     FOR i:=help^.y1 TO help^.y2 DO
       move(help^.scr[(i-help^.y1)*dx+1],
            mem[basis_screen:(pred(i)*help^.zl SHL 1 + x0)],
            dx);

   { Speicherbereich freigeben, Verweise aktualisieren }
   IF help^.pre=NIL THEN screen_pointer:=NIL
                    ELSE screen_pointer:=help^.pre;
   freemem(help,sizeof(help^.pre)+5*sizeof(help^.x1)+dx*dy)

END; { RESTORE_WINDOW }
```

Bild 1 Prozeduren zum Sichern und Rückspeichern von Bildschirmbereichen

```
PROGRAM Demo_fuer_SAVE_WINDOW_und_RESTORE_WINDOW;

VAR   i   :integer;

(*$I WDW-IBM.INC*)

PROCEDURE FRAME(x1,y1,x2,y2:byte);
CONST zeichen:ARRAY[1..6] OF char
             =(#218,#191,#192,#217,#196,#179);
VAR   i        :byte;
BEGIN
  window(1,1,bss,bsz);
  gotoxy(x1,y1);write(zeichen[1]);
  FOR i:=succ(x1) TO pred(x2) DO write(zeichen[5]);
  write(zeichen[2]);
  FOR i:=succ(y1) TO pred(y2) DO
  BEGIN
    gotoxy(x1,i);write(zeichen[6]);
    gotoxy(x2,i);write(zeichen[6])
  END;
  gotoxy(x1,y2);write(zeichen[3]);
  FOR i:=succ(x1) TO pred(x2) DO write(zeichen[5]);
  write(zeichen[4]);
  window(succ(x1),succ(y1),pred(x2),pred(y2));
  IF y2-y1>2 THEN clrscr
  ELSE BEGIN
    gotoxy(1,1);write('':pred(x2-x1),#13)
  END
END; { FRAME }

BEGIN { Demo.. }

  window(1,1,bss,bsz);
  screen_pointer:=NIL;      { Zeigervariable initialisieren }

  { in erster Bildebene arbeiten }
  SAVE_WINDOW(1,1,bss,bsz);
  clrscr;
  FOR i:=1 TO 300 DO write('Ebene 1  ');

  { in zweiter Bildebene arbeiten }
```

```
    SAVE_WINDOW(10,5,bss-10,bsz-5);
    FRAME(10,5,bss-10,bsz-5);
    FOR i:=1 TO 200 DO write('Ebene 2    ');

    { in dritter Bildebene arbeiten }
    SAVE_WINDOW(15,7,bss-5,bsz-2);
    FRAME(15,7,bss-5,bsz-2);
    FOR i:=1 TO 200 DO write('Ebene 3    ');

    { Rueckspeicherung in
      umgekehrter Reihenfolge }
    delay(2000);RESTORE_WINDOW(true);
    delay(2000);RESTORE_WINDOW(true);
    delay(2000);RESTORE_WINDOW(true);
    delay(2000);window(1,1,bss,bsz);
    clrscr

END. { Demo.. }
```

***Bild 2** Anwendungsbeispiel der Prozeduren aus Bild 1*

positionen in Bild 1 als Offset zur Basisadresse des Bildwiederholspeichers.
Bei der Manipulation des Videomoduls ist zu beachten, daß die Prozeduren in der vorliegenden Form nur auf der Bildschirmseite 0 arbeiten.
In den Quelltext ist lediglich die Einbindung der in Bild 1 aufgeführten Programmzeilen mittels der Includ-Option des Compilers zu organisieren (siehe auch Bild 2). Vor dem ersten Aufruf der Prozedur SAVE_WINDOW muß die Variable *screen_pointer* mit dem Wert *nil* initialisiert werden. Die Prozeduren SAVE_WINDOW und RESTRORE_WINDOW dürfen nur paarweise verwendet werden, eine Verletzung dieser Regel (oder ungenügender Speicherplatz im Heap) führt zur Ausgabe einer Fehlermeldung und zum Programmabbruch. Ebenso muß der Heap vor dem Aufruf von RESTORE_WINDOW auf den gleichen Stand wie unmittelbar nach dem entsprechenden Aufruf von SAVE_WINDOW gebracht werden. Gleiches gilt für den Bildschirmmodus sowie die aktuelle Bildschirmseite.
Bild 2 zeigt in einem kurzen Demonstrationsbeispiel die Anwendung der Prozeduren für die Programmierung von Dialogfenstern, die vom Bildschirm auch wieder verschwinden, wenn sie nicht mehr benötigt werden.
Die dargestellte Lösung ist in vielerlei Hinsicht ausbaufähig, man sollte jedoch prüfen, inwieweit bei komplexeren Anforderungen professionellere Programmierwerkzeuge günstiger einzusetzen sind.

Literatur

/1/ Geiler, J.; Wermann, M.: Der IBM PC und seine Kompatiblen. Mikroprozessortechnik, Berlin 2 (1988) 8, S. 234

KONTAKT

VEB Kombinat ILKA Luft- und Kältetechnik, Stammbetrieb Forschung und Technik Dresden, Bertold-Brecht-Allee 20, Dresden, 8019; Tel. 3 49 85 61

Mittel und Methoden der Künstlichen Intelligenz Teil 3

Jörg Schmidt
Technische Universität Karl-Marx-Stadt, Sektion Informatik

Suchstrategien in Spielsituationen

Spielsituationen in Graphen – MINIMAX

Wir betrachten Spiele zwischen zwei Spielern mit Ausschluß von zufälligen Einwirkungen (Glücksspiele) mit vollständiger Information. Daher gibt es außer dem Gegenzug des Partners, dessen Möglichkeiten bekannt sein müssen, keine unbekannten Faktoren für das System. Ziel ist eine Gewinnstrategie.
Wir betrachten als erstes ein Spiel von MIN und MAX (unsere beiden Spieler), die eine Anzahl von 7 Streichhölzern vor sich haben, und jeder, der am Zug ist, hat den Haufen, bzw. dann einen der Haufen, in ungleiche Bestandteile zu zerlegen. In Bild 17 ist der Spielverlauf dargestellt.
MAX : 5,2 bedeutet, MAX ist am Zug und hat zwei Mengen von Streichhölzern mit 5 bzw. 2 Stück vor sich. Wie die fettgedruckten Pfeile verdeutlichen, gibt es eine Gewinnstrategie für MAX unabhängig davon, wie MIN das Spiel beginnt. Hierbei sind gewisse Analogien zu UND/ODER-Graphen zu sehen. Nachfolgeknoten von MAX, also MIN-Knoten, stellen ODER-Knoten dar. Es reicht aus, wenn einer davon auf der Gewinnstrategie liegt. Andersherum sieht es mit den MAX-Knoten aus. Da ungewiß ist, welchen davon MIN auswählt, müssen alle auf der Gewinnstrategie liegen, bilden also eine Menge von UND-Knoten.
Das Spiel mit den Streichhölzern läßt sich noch recht einfach in einem solchen UND/ODER-Graph darstellen. Schwieriger wird es jedoch bei komplizierteren Brettspielen, wie Dame oder Schach. Sind es beim Damespiel etwa 10^{40} zu entwickelnde Knoten, so sind es beim Schach gar 10^{120}. Bei einer Generationsrate von 10 Knoten/Nanosekunde (das entspricht etwa 10 Giga-LIPS = 10^{10} Logical inferences per second; und das ist unvorstellbar für heutige Computer) würde ein vollständiger Graph für das Schachspiel eine Rechenzeit von 3×10^{100} Jahrhunderten erfordern.
Es ist daher sehr schnell einzusehen, daß das nicht der Weg sein kann, um die Gewinnstrategie zu finden. Wir geben uns deshalb damit zufrieden, einen guten ersten Zug zu finden. Das ist ständiges Ziel unseres Spielers. Die Suche nach diesem guten ersten Zug kann durch eine Reihe von Limitierungen eingeschränkt werden (Zeit, Speicherraum, Suchtiefe). Wenn die Suche bis zu diesen Grenzen fortgeschritten ist, erhalten die Blattknoten des Baumes (die, die keine Nachfolgeknoten besitzen) einen Wert zugeordnet, der im Zusammenhang mit der Güte der entsprechenden Spielsituation stehen sollte. Es könnte dies ein positiver Wert für einen für MAX günstigen Zug und ein negativer Wert für einen für MIN günstigen Zug sein. Dann würde MAX, wenn er vor der Zugauswahl steht, den Knoten mit dem maximalen Wert (daher auch die Namen unserer beiden Spieler) wählen und umgekehrt MIN den Knoten mit dem minimalen Wert. Zuvor müssen allerdings die Werte der Blattknoten an ihre Elternknoten weitergegeben werden, von diesen wieder an die Elternknoten usw. Das geschieht auf folgende Weise: Ein MAX-Elternknoten erhält den Wert des Nachfolge-MIN-Knotens mit maximalem Wert; umgekehrt für MIN. Das wird bis zu den Nachfolgern des Anfangsknotens fortgeführt, von wo aus dann entsprechend ausgewählt werden kann. Es wäre natürlich auch denkbar, bereits für die Nachfolgeknoten des Anfangsknotens dieselben Berechnungen anzustellen wie für die Blattknoten, um die entsprechenden Werte zu ermitteln. Man kann jedoch davon ausgehen, daß die von den Blattknoten zurückgegebenen Werte, da sie ja schon weiter im Spiel voraus *gedacht* haben, größeren Informationsgehalt besitzen.

Alpha-Beta-Suche

Bild 18 zeigt einen durch das soeben beschriebene MINIMAX-Verfahren erhaltenen Spielbaum. Die Quadrate stellen MAX-Knoten, die Kreise MIN-Knoten dar, und die Werte in diesen Kreisen bzw. Quadraten sind die durch die MINIMAX-Prozedur ermittelten. Es zeigt sich, daß ein Teil der Knoten für die Suche völlig unbedeutend ist und weggelassen werden kann. Bei einer Tiefensuche würden die Knoten in der Reihenfolge untersucht werden, wie sie in Bild 18 numeriert worden sind. Wenn die Konfigurationen 4. und 5. untersucht worden sind, erhalten wir für 3. den Wert 4; 2. ist ein MIN-Knoten, wählt demnach den Minimalwert unter seinen Nachfolgeknoten. Dieser kann nur kleiner als 4 werden. Wenn wir 6. untersuchen und feststellen, daß bereits 7. den Wert 5 liefert, also 6. mindestens den Wert 5 hat, braucht 8. gar nicht mehr untersucht werden. Ebenfalls braucht der ganze Zweig, der mit dem Knoten 13. verbunden ist, nicht betrachtet werden. Durch dieses Beschneiden des MINIMAX-Spielbaumes ergeben sich enorme Zeiteinsparungen. Bei einem Baum mit n Blattknoten brauchen durch dieses *Beschneiden* etwa nur $2 * \sqrt{n}$ Blattknoten untersucht werden. Die Erwartungswerte für MAX und MIN wurden bei ersten Untersuchungen zu diesem Verfahren mit Alpha und Beta bezeichnet. Knoten 1. könnte man dann einen Alpha-Wert von 4 zuordnen. Alle Zweige, die einen geringeren Wert haben, sind uninteressant. Wenn wir

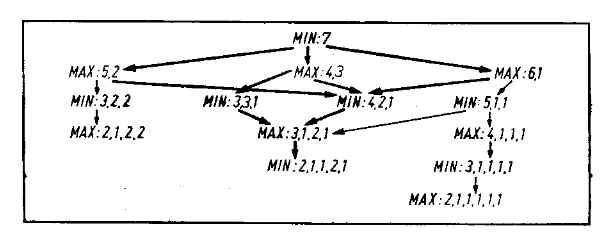


***Bild 17** Spielverlauf für MIN und MAX*

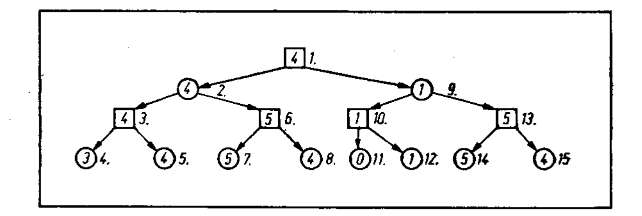


***Bild 18** Spielbaum mit MINIMAX*

von einem beliebigen Knoten aus mit dem Tiefenverfahren suchen, erhalten wir, wenn wir das erste Mal die maximale Tiefe erreichen, den Alpha-Wert für MAX-Knoten und den Beta-Wert für MIN-Knoten, die jeweils untere bzw. obere Schranke der zu erwartenden wirklichen Werte sind.

Expertensysteme

Der Begriff des Expertensystems ist in den letzten Jahren überall in der Welt zum Schlagwort geworden. Sowohl in der Fachliteratur zur Informatik, Computertechnik und Künstlichen Intelligenz als auch in der Wirtschaft, der Politik und der Tagespresse.

Aufbau eines Expertensystems

Ein Expertensystem ist erst einmal ein Computerprogramm. Dieses Computerprogramm dient einem Experten (einem Menschen) in einem begrenzten Problemkreis der objektiven Realität (einer Mikrowelt). In dieser Mikrowelt verfügt das Expertensystem über das nötige Wissen, das sich in einer Wissensbasis befindet. Es kann aber ebenfalls dazu dienen, daß ein *Nichtexperte* (ein Mensch) einen Experten (diesmal das Computerprogramm) um Rat fragt.
Bild 19 zeigt einen möglichen Aufbau eines Expertensystems. Die einzelnen Komponenten verdeutlichen schon die Aufgaben, die ein solches Expertensystem zu lösen hat. Über die Dialogkomponente ist es mit dem Nutzer verbunden. Hier ist anzustreben, über Möglichkeiten der Sprach- und Bildverarbeitung und der Computergrafik diese Schnittstelle immer mehr der Begriffswelt des menschlichen Nutzers zu nähern. Die Problemlösungskomponente leitet anhand der gegebenen Fakten und Regeln (also anhand des Wissens) logische Schlüsse ab. Die Erklärungskomponente sollte in der Lage sein, solche Prozesse dem Nutzer verständlich zu machen. Und die Wissenserwerbskomponente dient schließlich dazu, die Wissensbasis zu erweitern.
In der Welt existieren eine Reihe solcher Systeme zur Unterstützung der Expertentätigkeit. Sie sind eigentlich in allen Aufgabenbereichen einsetzbar, angefangen von der medizinischen Diagnose bis hin zu Hilfestellungen bei VLSI-Entwürfen bzw. als Bestandteil in CAD- und CAE-Systemen. Wie gesagt, Expertensysteme unterstützen die Handlungsfähigkeit derer, die mit ihnen arbeiten, sind aber selbständig nicht zu eigenen Handlungen fähig. Sie sind eine Art *Wissensverstärker* für ihren Nutzer, entlasten diesen dadurch aber nicht von seiner Verantwortung beim Treffen von Entscheidungen. Das ist in moralisch-ethischer Hinsicht auch besonders beim Einsatz in der Medizin zu beachten, spricht aber keineswegs gegen diesen Einsatz.
Neben einer Reihe von Ideen, wie Expertensysteme zu konstruieren sind, haben sich zwei grundlegende Schulen herausgebildet:
- Die logische Programmierung (z. B. Prolog) mit Fakten und Regeln in Form von prädikatenlogischen Klauseln, und
- die regelbasierte Schule (z. B. OPS5), bei der Regeln in Form von Bedingung-Aktion-Paaren festgeschrieben sind und Fakten als strukturierte Objekte (beispielsweise Frames) dargestellt sind. Diese werden durch die Regeln getestet und modifiziert.

Durch die Vielfältigkeit der möglichen Einsatzgebiete kann vor dem Informatiker oder dem *Wissensingenieur* nicht die Aufgabe stehen, Hunderte oder Tausende von solchen Systemen zu entwickeln, sondern wünschenswert ist es, Programmierumgebungen (sogenannte Shells) für Expertensysteme zu schaffen, die in vielen Gebieten einsatzfähig sind.

Expertensysteme erfordern neue Computerarchitekuren

Mit der Entwicklung von Expertensystemen entstanden eine Reihe von Problemen, die einerseits die Entwicklung hemmen, andererseits offene Fragen sind. Zu diesen hemmenden Faktoren gehören Verkaufsstrategien westlicher Firmen. Es scheint sich gegenwärtig in der Welt so zu gehören, daß Steuerungen u. ä. zusammen mit Expertensystemen verkauft werden müssen, um noch das große Geld machen zu können. Und der Begriff des Expertensystems ist dann nur noch ein anderes Wort für Computerprogramm. Das hat nichts mehr mit Künstlicher Intelligenz zu tun und führt auch zu einem gewissen Desinteresse der KI-Forscher an der Entwicklung von Expertensystemen. Andererseits läßt diese Tendenz die Annahme zu, daß das Problem der Expertensysteme längst gelöst sei, und es keine Hürde ist, beliebiges Expertenwissen in so ein System zu pressen. Es führt weiterhin zu einer Unterschätzung der für solche Systeme notwendigen Computertechnik, außerdem zu dem Bestreben, Mittel und Methoden der Künstlichen Intelligenz einzusetzen, wo es herkömmliche Programmierungstechniken bisher auch und sogar besser getan haben; beispielsweise wird ein in Prolog geschriebener Sortieralgorithmus noch lange nicht zu einem Expertensystem für das Sortieren!
Diese Tendenzen wirken auf das Erschließen der offenen Probleme und Möglichkeiten der Expertensysteme gewiß nicht förderlich, scheinen aber international zur täglichen Praxis zu werden.
Nun zu den offenen Problemen:
Als erstes gilt es hier die Frage der Grundlagenforschung zu betrachten. Die Erforschung der Mittel und Methoden der Künstlichen Intelligenz reicht sehr weit zurück und hat auch schon eine Reihe in sich abgeschlossener Ergebnisse gebracht, wie aus den vorangegangenen Teilen dieses Beitrages im MP 6 und 9/1989 zu ersehen ist. Es gibt allerdings keine allgemeingültige Theorie darüber, welche Methode wann auszuwählen ist. Ähnlich sieht es mit der Frage der Art der Wissensrepräsentation aus.
Welche ist für welchen Fall geeignet, wie werden sie ineinander umgewandelt (Bilder in Prädikatenlogik usw.)?
Ein weiteres nicht zu unterschätzendes Problem ist die Menge an Wissen, über das ein Expertensystem verfügen sollte.
Einerseits stehen Forderungen nach einem größeren Wissensumfang. Das U.S. Department of Defense's Advanced Research Projects Agency fordert für 1990 Expertensysteme mit 30 000 Regeln und einer Geschwindigkeit von 12 000 productions per second; das Japanische Projekt zur Fünften Computergeneration geht von mehr als 10 000 Klauseln bei einer Geschwindigkeit von 50 Mega bis 1 Giga-LIPS ($50 \cdot 10^6 \ldots 10^9$ LIPS) aus; andere Angaben gehen noch weiter. Demgegenüber stehen skeptische Meinungen von einer Reihe von KI-Forschern. Diese bezweifeln die Möglichkeit der Verwaltung von größeren Systemen bis hin zu der Ansicht, daß bei mehr als 8 000–10 000 Regeln Verluste durch Inkonsistenz entstehen können.
Was die Erweiterung der Kapazitäten der Expertensysteme anbelangt, so führt diese gezwungenermaßen zu einem Umbruch in der Computerarchitektur überhaupt. Die Geschwindigkeitssteigerung läßt sich nur durch Parallelverarbeitung einerseits und Compileroptimierungen andererseits erreichen. Weiterhin sollten diese Architekturen über einen größeren linearen (d. h. nicht segmentierten) Adreßbereich verfügen, damit auch umfangreichere Programme lauffähig sind.
Damit sind der Stand und einige Tendenzen der internationalen Entwicklung auf diesem Gebiet dargelegt.

Literatur

/1/ Nilsson, N. J.: Principles of Artificial Intelligence. Springer Verlag Berlin, Heidelberg, New York 1982
/2/ Alty, J. L.; Coombs, M. J.: Expert Systems Concepts and Examples. The National Computing Centre Limited 1984
/3/ Steels, L.: The Deepening of Expert Systems. AI Communications, Vol. O, No. 1, August 1987
/4/ van de Riet, R. P.: Problems with Expert Systems? Future Generation Computer Systems, Vol. 3, No. 1, February 1987

KONTAKT

Technische Universität Karl-Marx-Stadt, Sektion Informatik, PSF 964, Karl-Marx-Stadt, 9010; Tel. 66 85 21

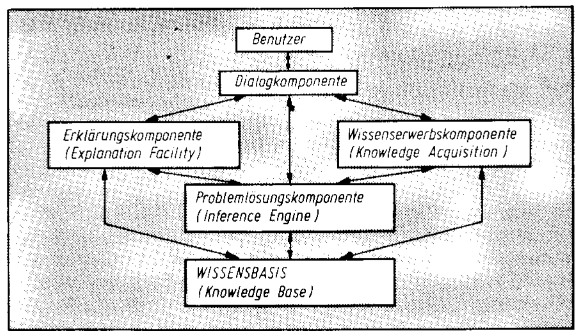


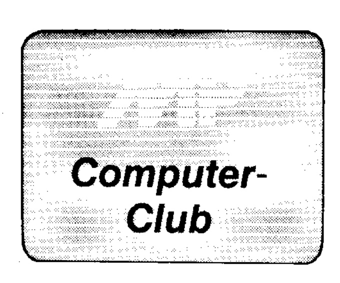
Bild 19 Architektur eines Expertensystems

Kleines Lexikon der Mikrorechentechnik

P
wie Priorität



Zeichnung: Dahmen



Zweiter Drucker am A 7100/7150

Hartmut Voigt, Finsterwalde

In einigen Anwendungsfällen ist es notwendig, über einen Personalcomputer zwei Drucker zu bedienen, zum Beipiel wenn im ständigen Wechsel auf Formularen und formlosem Papier gedruckt werden soll. Im folgenden wird eine Lösung dieses Problems am A 7100 unter dem Betriebssystem SCP 1700 beschrieben.

Mit dem Betriebssystem SCP 1700 läßt sich standardmäßig ein Drucker über Centronics-Schnittstelle betreiben. Soll zusätzlich ein zweiter Drukker mit V.24-Eingang angeschlossen werden, so sind folgende Änderungen im Betriebssystem notwendig: Das INITSI3-Byte am Anfang des Quelltextes muß dem jeweiligen Druckertyp angepaßt werden. Für den Epson LX 86 lautet es zum Beipiel 10101111B. Der Drucker kann nach dem Generieren des Systems durch Ansprechen des Ausgabegerätes CRT bedient werden. Im Anwenderprogramm erreicht man das problemlos mit dem Setzen des I/O-Bytes auf 40H mittels BDOS-Funktion 8. Der genaue Wert des INIT-Parameters muß dem Handbuch des jeweiligen Druckers entnommen werden und wird nach den dazugehörigen Kommentaren im BIOS-Quelltext codiert. Das Anschlußkabel wird nach den Angaben in /1/ und im Handbuch des Druckers hergestellt. Man wird jedoch immer bestrebt sein, zwei Drukker gleichen Typs einzusetzen. Ein zweiter V.24-Ausgang läßt sich aber nur mit erhöhtem Aufwand, unter Einsatz eines aktiven Adapters, realisieren.

Stehen zwei Drucker mit Centronics-Eingang zur Verfügung, stellt sich der Fall wesentlich einfacher dar. Es müssen lediglich ein passiver Adapter gebaut und einige einfache Änderungen am Betriebssystem ausgeführt werden. Als Ausgang wird die IFSP-Schnittstelle benutzt. Hier muß allerdings beachtet werden, daß die Ausgänge über Treiberstufen arbeiten und demzufolge negiert an den Steckkontakten anliegen. Dieses Problem wird softwareseitig gelöst.

Der Adapter

Um eine Änderung des Druckeranschlußkabels zu vermeiden, wird ein einfach aufgebauter Adapter benutzt. Dazu werden jeweils eine Steckerund eine Buchsenleiste C 25 (Cannon) benötigt. Die Kontakte der Leisten werden entsprechend Bild 1 durch dünne Schaltdrähte verbunden. Nach der Erprobung wird ein Blechstreifen als Kapselung um die Leisten gelegt. Ein Stück dünne Plastfolie isoliert die Lötkontakte. Wie die mechanische Verbindung der beiden Leisten erfolgt, richtet sich nach den handwerklichen Möglichkeiten. In unserem Fall wurde ein Stück Blech an den Kanten 2 mm abgewikkelt, in den Kragen der Leisten gelegt und mit einer Schraube an der Kapselung befestigt. Eine andere Möglichkeit wäre das Ausgießen des ausgetesteten Adapters mit Epoxidharz.

Wer den Aufbau des Adapters umgehen möchte, kann natürlich auch das Anschlußkabel entsprechend modifizieren. Es muß dann jedoch eine Verwechslung mit dem Originalkabel vermieden werden.

Das Betriebssystem

Änderungen am Betriebssystem sind an drei Stellen notwendig:

- Die Alternativkonstante LISTIFSP am Anfang des Quelltextes wird auf TRUE gesetzt.
- Der Statustest (nach der Marke IFSPST:) wird nach Bild 2 geändert.
- Die Zeichenausgabe (IFSPOUT:) ist nach Bild 3 neu in das Programm aufzunehmen.

Die Initialisierung des IFSP-Ausgangs wird nicht geändert. Das schon genannte Problem, daß alle Pegel negiert ausgegeben werden, bedeutet, daß die sonst zu verwendenden Handshakesignale BSTROBE und BRDY durch entsprechende Nachbildungen in der Ausgabe ersetzt werden müssen. Dieses Verfahren bringt

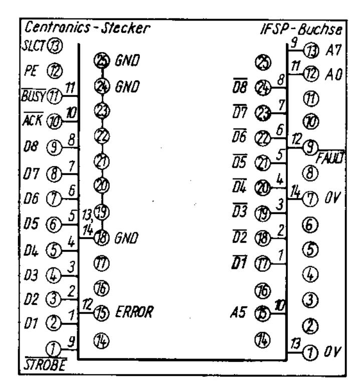


Bild 1 Adapter für IFSP nach Centronics-Umbelegung

```
LISTST_UL1:
        IF      NOT LISTIFSP
;-----------------------------------------------------
        JMPS    IFSSOUTST
;-----------------------------------------------------
        ENDIF
; Bis hier Original-Betriebssystem

IFSPST:
        MOV     DX,IFSP+2       ;Adresse Port A
        IN      AL,DX
        TEST    AL,020H         ; /ACK-Test
        JNZ     CONST_NO1       ; Drucker nicht bereit
        AND     AL,1            ; /BUSY-Test
        JMPS    AUSW            ; zentrale Auswertung

; Ende IFSPST-Zusatz
;-----------------------------------------------------
```

Bild 2 Statusprüfung der IFSP-Schnittstelle am Centronics-Drucker

```
IFSPOUT:
        CALL    IFSPST
        JZ      IFSPOUT
        IN      AL,DX
        TEST    AL,4            ; ANY ERROR?
        JZ      UL1NOE          ; IF Z NO
        CALL    LSTERR          ; PRINT ERROR MESSAGE
        JMPS    IFSPOUT         ; TRY AGAIN
; bis hier Original-Betriebssystem
;-------------------------------------------------------------
; Ausgabe am IFSP-Interface zu einem Centronics-
; drucker
; Port A: Statusinformationen, Port B: Daten
; A7- /STROBE, A5- /ACK, A4- PE, A3- /ERR, A0- /BUSY
; A6, A2, A1 nicht verdrahtet

UL1NOE:
        PUSH    CX
        AND     CL,7FH          ; nur ASCII
        NOT     CX              ; negieren wegen
                                ; Treiberstufe nach PIO
        MOV     AL,CL
        POP     CX
        MOV     DX,IFSP+4       ; Port B
        OUT     DX,AL
        DEC     DX
        DEC     DX              ; DX: jetzt Port A
        MOV     AL,080H
        OUT     DX,AL           ; /STROBE ein
        MOV     AL,0
        OUT     DX,AL           ; /STROBE aus
        RET

;       JMP     OUT     Sprung zur ehemaligen Ausgabe

;Ende UL1NOE-Einbau
```

Bild 3 Einzelzeichenausgabe über IFSP-Schnittstelle

```
program print_test;
var ep : string[255];

procedure outport(c: char);  {Einzelzeichenausgabe      }
const   IFSP = $308 ;        {Portadresse IFSP-Status}
        count= 1024 ;        {Dauer Warteschleife       }
var     t     : integer ;
        m     : char ;
function printer_ready:boolean;   {Statustest Drucker }
begin
  printer_ready:=true;
  if (port[IFSP] and $20 ) <> 0
    then exit;                    { /ACK-Test           }
  if (port[IFSP] and  1 )  <> 0
    then exit;                    { /BUSY-Test          }
  printer_ready:=false;           { sonst Fehler        }
end;
begin
  t:=port[IFSP];               { Abholen Fehldaten       }
  repeat                       { Wiederholen bei Fehler }
    t:=0;
    while t<count do           { Warten auf Druckstatus }
    begin                      { oder Zeitende           }
      t:=t+1;                  { Zeitzaehler             }
      if printer_ready then
      begin
        port[IFSP]:=0;         { Statusport loeschen     }
        port[IFSP+2]:=not byte(c);  { Ausgabe Zeichen}
        port[IFSP]:=$80;       { Strobe setzen           }
        port[IFSP]:=0;         { Strobe ruecksetzen      }
        exit                   { zurueck nach Ausgabe    }
      end
    end;                       { Zeit ueberschritten     }
    write('Drucker nicht bereit. <r> - Wiederholen;');
    write(' <i> - naechstes Zeichen; <c> - Abbruch');
    repeat
      read(kbd,m); m:=upcase(m)
    until m in ['R','I','C'];
    write(m);
    if m='C' then halt         { Abbruch des Programms   }
  until m='I'                  { Ausgabe unterdrueckt    }
end;

begin                          { Testprogramm            }
  lstoutptr:=ofs(outport);     { Umleiten der List-      }
                               { ausgabe auf nutzer-     }
                               { definierte Routine      }
  repeat
    ep:='';
    readln(ep);
    writeln(lst,ep)
  until ep=''
end.
```

Bild 4 Ausgabetreiber für Turbo-Pascal

aber keinen Nachteil in der Anwendung mit sich. Die Kommentare in den Bildern 2 und 3 sollen zur Erklärung genügen. Um das Auffinden der Stellen im Quelltext zu erleichtern, werden einige Zeilen vor und hinter den Änderungen mit abgebildet.
Nach der Übersetzung und dem Start des Betriebssystems kann der Drukker mittels STAT LST: = UL1: und anschließender Hardcopyfunktion durch Control+P getestet werden.
Zur Einbindung des Druckers in das Anwenderprogramm gilt das schon zur V.24-Schnittstelle Gesagte. Das I/O-Byte wird jetzt auf 0C0H gesetzt.
Wem das Umstellen des Betriebssystems zu aufwendig ist oder wer Skrupel hat, hier eine Änderung durchzuführen, für den sei noch eine andere Lösung dargestellt. Im Turbo-Pascal ist es beispielsweise möglich, die Ausgabetreiber logischer Geräte auf selbstgeschriebene Routinen umzuleiten. Bild 4 zeigt ein Testprogramm. Die Prozedur OutPort stellt diesen Treiber dar. Die darin eingebrachte Funktion PrinterReady prüft den Status des Druckers. Wird ein anderer Drucker als der hier verwendete LX 800 benutzt, muß die Konstante Count eventuell verändert werden. Sie ist von der Ausgabezeit eines Zeichens abhängig. Um den Treiber einzuschalten, ist die erste Zeile des Hauptprogramms LstOutPtr:=Ofs (Outport) notwendig.

Literatur

/1/ Betriebsdokumentation A 7100, Band 1. Rechner und Geräte. VEB Robotron-Projekt Dresden, 1986

Dienstprogramme für SCP 1700

Frank Isekeit
Deutsche Reichsbahn, Ingenieurbüro für Rationalisierung des Eisenbahnbaues

In dem folgenden Artikel werden zwei Dienstprogramme zur Verbesserung der Arbeit mit der Speicherdiskette RAM-Disk unter SCP 1700 vorgestellt. Die Programme ermöglichen es, ohne RESET den reservierten Speicherplatz freizugeben als auch die RAM-Disk neu einzurichten und zu initialisieren.

Die relativ umständlichen Handlungen zur Vorbereitung der Nutzung der Speicherdiskette führten zu Überlegungen, dieses Hilfsmittel beim Systemstart ohne Zutun des Nutzers grundsätzlich einzurichten /1/, /2/. Das ist – neben der Erleichterung der Bedienung – auch eine wesentliche Voraussetzung zur Arbeit mit einem Kaltstartkommando. Einige bekannte Programme (u. a. TABCALC) sind auf Geräten mit 512 KByte Hauptspeicher aber nur lauffähig, wenn keine Speicherdiskette eingerichtet ist. In einem solchen Fall ermöglicht das Programm CLRDSK, den für die RAM-Disk reservierten Speicherbereich freizugeben. Das Programm GENDSK dient dagegen zum Einrichten und Initialisieren der RAM-Disk. Beide Programme sind für das Betriebssystem SCP 1700 (Version 2.2 und 3.0) einsetzbar. Bei anderen Betriebssystemen ist die Eignung zu überprüfen.
Die Programme sollen im folgenden kurz vorgestellt werden. Hinweis: Bei beiden Programmen ist – im Hinblick auf die Nutzung in SUBMIT-Dateien – keine Rückfrage an den Bediener als Schutzmaßnahme bei versehentlichem Aufruf vorgesehen.

Das Programm CLRDSK

Der Quelltext des Programms ist in Bild 1 wiedergegeben, es besitzt keine Besonderheiten. Eine Mitteilung an den Bediener erfolgt nicht. CLRDSK setzt im Tabellenbereich des Betriebssystems die Basis-Segmentadresse der RAM-Disk auf Null (wird von SCP 1700 als *nicht vorhandene* RAM-Disk interpretiert) und vergrößert den TPA-Bereich um den sonst reservierten Bereich.
Da CLRDSK keine Schreiboperationen in dem vorgesehenen Hauptspeicherbereich ausführt, kann man bei versehentlichem Aufruf gespeicherte Dateien retten. Dazu sind die RESET-Taste und sofort nach dem ersten Kontrollton die BREAK-Taste zu betätigen /3/. Beim Betriebssystem nach /2/ sind keine weiteren Maßnahmen erforderlich, beim Original-Betriebssystem ist die RAM-Disk einzurichten und nicht zu initialisieren. Wurde das Betriebssystem nach /1/ geändert, ist das Retten der Dateien allerdings nicht mehr möglich.

Das Programm GENDSK

Bild 2 zeigt den Quelltext des Programmes, das eine Reihe von Prüfungen durchführt, die im wesentlichen den vom Betriebssystem vor Einrichten der RAM-Disk vorgenommenen entsprechen. Das betrifft im einzelnen:

- Sind mindestens 5 Diskettenlaufwerke definiert?
- Ist der Diskettenparameterblock für physisches Laufwerk 4 vorhanden?
- Ist das physische Laufwerk 4 vom Typ K5600.20?
- Beträgt die Hauptspeichergröße mindestens 512 KByte?

Ist eine dieser Bedingungen nicht erfüllt, kann keine RAM-Disk eingerichtet werden. In diesem Fall gibt das Programm eine Fehlermeldung aus.
Bei erfolgreicher Abarbeitung wird nach einer entsprechenden Meldung die Speicherdiskette initialisiert und zurückgesetzt, so daß deren Nutzung ohne vorherige Eingabe von CTRL-C erfolgen kann (wichtig bei Abarbeitung von GENDSK in Kommandodateien).
Der Aufruf von GENDSK bei bereits vorhandener RAM-Disk bewirkt ein Initialisieren derselben.

Das Übersetzen der Programme

Beide Programme nutzen das 8080-Speichermodell und entsprechen in ihrem Aufbau dem in /4/ gezeigten Beispiel.
Die nachstehenden Erläuterungen wurden im wesentlichen /4/ und /5/ entnommen und wenden sich, entsprechend dem Charakter der Geräte als Arbeitsplatzcomputer, in erster Linie an den Nutzer mit geringerer Erfahrung.
Zunächst sind die Quelltexte der Programme einzugeben und in den Dateien CLRDSK.A86 und GENDSK.A86 abzuspeichern. Wird dabei ein Textverarbeitungssystem genutzt, muß in einem zum Erstellen von Programmen geeigneten Modus gearbeitet werden. Die weitere Vorgehensweise sei am Beispiel von CLRDSK gezeigt.

```
        title   'clrdsk'
        cseg
        org     100h
        mov     cl,32h
        mov     dx,offset bio
        int     224
        mov     ax,0104h
        mov     ds,ax
        mov     ax,6[bx]
        cmp     ax,0
        jz      exi
        xor     ax,ax
        mov     6[bx],ax
        mov     ax,3[bx]
        add     ax,4f80h
        mov     3[bx],ax
exi:    mov     cl,25h
        mov     dx,0010h
        int     224
        mov     cl,0
        mov     dl,0
        int     224
endcs   equ     $
        dseg
        org     offset endcs
bio     db      18,0,0,0,0
```

Bild 1 Quelltext des Programms CLRDSK

Zunächst übersetzt man den Assembler-Quelltext mit ASM86:
ASM86 CLRDSK ‹CR›
und generiert anschließend das Ladeformat:
GENCMD CLRDSK 8080 ‹CR›
Sollten bei der Arbeit von ASM86 Fehler gemeldet werden, kann man diese anhand der Listen-Datei CLRDSK.LST lokalisieren, die Quelldateien korrigieren und den Vorgang wiederholen.
Die nach dem Lauf vom GENCMD entstandene Datei CLRDSK.CMD ist bereits das lauffähige Programm.
Die Programme ASM86 und GENCMD gehören zum Softwarepaket des A 7100.

Literatur

/1/ Lennartz, M.: **Veränderungen des SCP 1700.** Mikroprozessortechnik 2 (1988) 4, S. 124
/2/ Herse, M.; Isekeit, F.: Änderungen am Betriebssystem SCP 1700. Mikroprozessortechnik 2 (1988) 6, S. 181
/3/ Datei DOKERG.TXT auf der Systemdiskette des A 7100
/4/ Arbeitsplatzcomputer A 7100, Anleitung für den Systemprogrammierer, Teil „Steuerprogramm SCPX", S. 12
/5/ Arbeitsplatzcomputer A 7100, Anleitung für den Programmierer, Teil „Debugger und Assembler", S. 7


```
;**********************************************************
;  Programm GENDSK - Einrichten und Initialisieren
;  der Speicherdiskette
;  (c) Ibr- Eb, F. Isekeit                      20.06.88
;**********************************************************
        TITLE   'GENDSK'
        CSEG
        ORG     100H
        MOV     CL,32H
        MOV     DX,OFFSET BIO
        INT     224
        MOV     AX,0104H
        MOV     DS,AX
        MOV     AL,9[BX]
        CMP     AL,5
        JL      ERR
        MOV     AX,6[BX]
        CMP     AX,0                    ;SPEICHERDISKETTE AKTIV?
        JNE     INI                     ;WENN JA NUR INITIALISIEREN
        MOV     AX,BX
        ADD     AX,86                   ;DPB- ADRESSE BERECHNEN
        MOV     SI,AX
        MOV     AX,[SI]
        OR      AX,AX                   ;DPB- ADRESSE VORHANDEN?
        JZ      ERR                     ;FEHLER WENN NICHT
        ADD     AX,17
        MOV     SI,AX
        MOV     AL,138
        CMP     [SI],AL                 ;TYP K5600.20?
        JNZ     ERR                     ;FEHLER, WENN NICHT
        MOV     AX,-2[BX]
        CMP     AX,8000H                ;HS > 512KBYTE?
        JB      ERR
        SUB     AX,4F80H
        MOV     6[BX],AX                ;DISK- SEGMENT
        MOV     AX,3[BX]
        SUB     AX,4F80H
        MOV     3[BX],AX                ;SEG- LENGTH
INI:    MOV     AX,6[BX]
        MOV     DS,AX
        MOV     CX,2000H
        XOR     BX,BX
        MOV     AX,0E5E5H
LO:     MOV     [BX],AX
        ADD     BX,2
        LOOP    LO
        MOV     CL,25H
        MOV     DX,0010H
        INT     224                     ;RUECKSETZEN LAUFWERK E:
        MOV     DX,OFFSET MSG1
        JMPS    MSG
ERR:    MOV     DX,OFFSET MSG2          ;FEHLERMELDUNG
MSG:    PUSH    CS
        POP     DS
        MOV     CL,09H
        INT     224
        MOV     CL,0
        MOV     DL,0
        INT     224
ENDCS   EQU     $
        DSEG
        ORG     OFFSET ENDCS
BIO     DB      18,0,0,0,0
MSG1    DB      0DH,0AH,0AH,'Speicherdiskette E: (306 KByte)'
        DB      ' eingerichtet',0DH,0AH,'$'
MSG2    DB      0DH,0AH,0AH,'Einrichten der Speicherdiskette fuer'
        DB      ' dieses Betriebssystem nicht moeglich !',0dh,0ah,'$'
        end
```

Bild 2 Quelltext des Programmes GENDSK

DCP für 16-Bit-PCs ohne Festplatte

Für 16-Bit-PCs (A 7150, EC 1834), die über kein Festplattenlaufwerk verfügen, wird ein optimaler Arbeitsablauf durch die Nutzung verschiedener Prozeduren (Batchdateien) unter dem Betriebssystem DCP ermöglicht. Voraussetzung für diese Arbeitsweise ist ein Computer mit zwei Floppy-Disk-Laufwerken. Beim Start dieser Prozeduren ist zu beachten, daß keine residente Benutzeroberfläche (Norton-Commander, QDOS, XTREE) aktiv sein darf, da bestimmte Parameter in der Umgebung (Environment) des Befehlsprozessors (COMMAND.COM) definiert und an die Folgeprozeduren übergeben werden müssen.
Jeder Nutzer hat die Möglichkeit, sich eine persönliche Diskette mit hierarchischer Verzeichnisstruktur einzurichten, welche während der gesamten Arbeit am Computer in einem Diskettenlaufwerk verbleiben kann. Ständig benötigte Software bzw. Prozeduren werden beim Systemstart auf eine RAM-Diskette (Laufwerk variabel) ausgelagert und das zugewiesene Laufwerk in die Suchpfadliste eingetragen. Das freie Diskettenlaufwerk dient zur Aufnahme und zum Wechsel der benötigten (Standard-) Software, welche nur einmal am Computerarbeitsplatz vorhanden sein muß. Die Abspeicherung der nutzerspezifischen Daten und Programme erfolgt auf die vereinbarte Nutzerdiskette. Das entsprechende Laufwerk oder die zugewiesenen Pfade werden ebenfalls vorrangig in die Suchpfadliste des Environments eingetragen. Die Einstellung auf die Standardparameter des Betriebssystems wird durch eine entsprechende Endeprozedur gewährleistet.
Bei Zusendung einer formatierten Diskette (720 KByte) können unter Angabe des Computertyps die Prozeduren AUTOEXEC.BAT, FBDISK-.BAT, LOGON.BAT, LOGOFF.BAT einschließlich einer Kurzdokumentation angefordert werden.

VEB Papierverarbeitungswerk Zittau, Fachbereich Ökonomie/EDV, Pescheckstraße 25, Zittau, 8800; Tel. 2553, App. 54
Dittrich

Kopplung EMR – PC

Eine ökonomische Alternative zum Einsatz von Rechnern mit 16-/32-Bit Verarbeitungsbreite bieten Kopplungen vom EMRs mit PCs zur parallelen Meßwerterfassung. Die im Beispiel verwendete EMR-Platine (Basis U 8830) verfügt über 2 KByte externen EPROM (Editor/Debugger einschließlich modifizierter PUTCHAR- und GETCHAR-Routinen) und 4 KByte RAM. Je 8 der insgesamt 32-In-/Output-Leitungen ermöglichen die Kommunikation mit dem PC. Dabei werden die Ports % 2003 (Out) und % 2007 (In) mit der PC-PIO oder mit einer Porterweiterung des PCs verbunden. Die entwickelten PUTCHAR- und GETCHAR-Routinen ermöglichen die Programmierung des EMRs vom PC aus, wobei alle Vorteile der PC-Peripherie (Massenspeicher, Drucker usw.) genutzt werden. Die Programmierung kann sowohl im Tiny-Basic des U 8830 als auch im Maschinencode des U 8810 erfolgen. Ein komfortables PC-Programm (C 128) wurde dafür entwickelt. Eine Variante für U 880-Rechner ist in Vorbereitung.
Die übrigen je 24 In-/Output-Leitungen sind mit verschiedenen Meßgeräten (im Beispiel: Temperatur, Druck, Drehzahl/Drehmoment, Luftverbrauch) gekoppelt. Nach dem Start des EMRs erfolgen die vom PC unabhängige Meßwerterfassung und die Speicherung der Daten. Die Zwischenspeicherung von etwa 1 K × 16 Bit Meßwerten im RAM ist möglich. Mittels einer Synchronisierungsvorschrift erfolgt zu vom PC vorgegebenen Zeiten die Datenübernahme in den PC-RAM, wo die Weiterverarbeitung erfolgt.

Technische Hochschule Leipzig, Sektion Bauingenieurwesen, Lehrstuhl Arbeitsingenieurwesen, Karl-Liebknecht-Straße 132, Leipzig, 7030; Tel. 3928344
Bröcker/Sturm

Driver für IFSS-ZVE-Schnittstelle am A 7150

Es wird eine Lösung angeboten, einen Drucker mit IFSS-Schnittstelle am A 7150 unter DCP an der ZVE-IFSS-Schnittstelle betreiben zu können. Diese Schnittstelle ist im A 7150 mit Grafik nicht belegt, so daß keine funktionellen Einschränkungen auftreten. Man erhält sich zusätzlich die Möglichkeit, auf die ASP zu verzichten und diesen Steckplatz anderweitig zu verwenden.

Akademie der Wissenschaften der DDR, Institut für Kosmosforschung, BfSL, Rudower Chaussee 5, Berlin, 1199
Dr. Neumann

Nutzeroberfläche für 8-Bit-PCs

Die Arbeit mit Systemkommandos und Dienstprogrammen des PC 1715 und ähnlicher Rechentechnik ist wenig komfortabel. Deswegen entstand eine menügesteuerte Nutzeroberfläche für Personal- und Bürocomputer (PC 1715, BC 5120/30) unter den Betriebssystemen SCP und CP/A. Sie übernimmt die Dialogführung mit dem Nutzer und ist weitgehend bediensicher. Unter anderem werden auch Systemfehler wie eine offene Laufwerksklappe abgefangen. Integriert ist die gesamte Diskettenverwaltung (Kopieren, Löschen, Zurückholen von Dateien u. a.). Im aktuellen Directory können einzelne Files oder Filegruppen für bestimmte Operationen selektiert werden. Jede Datei kann angezeigt werden, dabei kann man vor- und zurückblättern und den Druck wahlweise zuschalten. Anwenderprogramme können abgearbeitet werden, nach deren Beendigung steht einem die Nutzeroberfläche wieder zur Verfügung. Das Arbeiten mit verschiedenen Userbereichen wird voll unterstützt. Das Programm ist selbsterklärend, zusätzlich ist eine Referenzkarte als WordStar-Datei vorhanden. Im Lieferumfang sind die Dateien **QUICK.COM** (Nutzeroberfläche), **QUICK.TXT** (Referenzkarte) und **QUICKLNG.TXT** (Nutzerbeschreibung) enthalten. Bei Zusendung einer formatierten Diskette (780 KByte) ist eine Demoversion erhältlich. Allen Nutzern wird ein kostenloser Update-Service angeboten, das heißt, bei Vorliegen einer neuen Version kann diese angefordert werden.

VEB Werkzeugmaschinenfabrik Aschersleben, Bereich 0, Wilslebener Straße 9–11, Aschersleben, 4320; Tel. 742130
Helmstedt/Kallas

Modulare Programmothek für den KC 87

Für den Bereich der polytechnischen Ausbildung in der Oberschule wurde ein Modulsystem auf der Basis des ROM-Erweiterungsmoduls 690 002.7 für den KC 85/1 und den KC 87 entwickelt. Dieses Modulsystem ermöglicht das Speichern von Basic-Programmen von bis zu 14,9 KByte auf EPROMs U 2716. Die unter diesem System bearbeitete „Grundsoftware für den Informatikunterricht der Klasse 9" ist nach dem Aufruf vom Modul äquivalent der von Kassette ladbaren Software nutzbar. Wesentliche Vorteile sind:
- Eliminieren der Ladezeit von Kassette (Programm steht durch Stecken des Moduls sofort im Computer zur Verfügung)
- Verhindern von Ladefehlern
- Angleich an die Ladegeschwindigkeit des A 5105.

Zur *Modularen Programmothek* gehört weiterhin eine Dokumentation, die alle programmcharakteristischen und fachspezifischen Daten, die für den Einsatz im Unterricht erforderlich sind, enthält. Des weiteren werden didaktisch-methodische Hinweise für die Arbeit des Lehrers mit der Software gegeben.

VEB Numerik „Karl-Marx", TN (Koll. Stark), PSF 947, Karl-Marx-Stadt, 9010
Nestler

Datenbankbaukasten db-Box

Immer häufiger verlangen Anwendungslösungen, dBase-Datenfiles zu nutzen, jedoch auf dBase aus verschiedenen Gründen zu verzichten. Das bedeutet, daß Nutzerprogramme in der Lage sein müssen, flexibel auf beliebig strukturierte dBase-Datenfiles zugreifen zu können. Darüber hinaus stehen die Forderungen nach
- Erhöhung der Software-Produktionseffektivität durch Nutzung von Basismodulen aus Baukästen
- Verbesserung von Softwareergometrie und Nutzungsbreite durch modulare Systeme

auf der Tagesordnung.
Diesen Forderungen entspricht ein Datenbankbaukasten für MS-DOS-Rechner des Kombinates Wälzlager und Normteile. Der Baukasten beinhaltet einzelne lauffähige Programme, die für eine Datenbankarbeit innerhalb von Anwendungslösungen geeignet sind. Sämtliche Module greifen auf beliebige dBase-Dateien zu. Ihre Arbeitsweise kann durch Schalter oder mittels Steuerdateien in weiten Grenzen modifiziert werden.
Die Programme wurden in Microsoft-Quick-Basic 4 programmiert. Für Interessenten stehen die Quelltexte der Basisprozeduren oder die Objektmodule zur Verfügung. Derzeit werden folgende Module angeboten: universeller Blättermodul, universeller Recherchemodul, Anhängen und Einfügen von Daten, Sortieren von Dateien, Editieren von Dateien, Editieren der „versteckten" Kopfinformationen, Umwandlung zwischen dBase II, III und Textdateien, Löschen von Daten, Erzeugen und Verändern von Datenstrukturen, spezieller DIR-Modul, Schützen und Entschützen von Dateien mittels Codes, universeller ASCII-Text-Lister, db-Box-Integrator (Menürahmen zur Arbeit mit den Modulen).

VEB Kombinat Wälzlager und Normteile, Betriebsteil Forschung und Entwicklung, Merseburger Straße 8, Leipzig-Rückmarsdorf, 7101
Dr. Löschke

Portabilität getypter Pascal-Dateien

Beim Erfassen und Verarbeiten großer Datenmengen besteht die berechtigte Forderung nach uneingeschränkter Datenportabilität. Unter SCP speichern getypte Pascal-Dateien in den ersten 4 Byte Satzanzahl und Satzlänge ab. Unter DCP tritt dies nicht mehr auf. Mit den von uns entwickelten SCP-DCP.BOX (Includedatei für DCP) und DCP-SCP.BOX (Includedatei für SCP/SCP 1700) lassen sich die Portabilitätsprobleme unkompliziert lösen. Das heißt, getypte Dateien auf SCP-Disketten (z. B. 780 KByte) können sofort unter DCP weiterverarbeitet werden. Natürlich vorausgesetzt, daß ein entsprechender SCP-Treiber (SCPDR.SYS) in die CONFIG.SYS eingebunden wurde. Notwendige Hinweise (Veränderung der RECORD-Vereinbarung u. a.) sind in der bereitgestellten SCP-DCP.DOK enthalten.

Institut für Forstwissenschaften Eberswalde, Organisations- und Rechenzentrum Potsdam, Koll. Neumann, Virchowstraße 39/41, Potsdam, 1591; Tel. 76941
Dr. Barciok

PETRI-Netz-Interpreter für Einchipmikrorechner

Petri-Netze gestatten es, Abläufe, die auch parallele Zweige enthalten können, klar und übersichtlich darzustellen. Solche Abläufe sind in der Steuerungstechnik häufig anzutreffen. Für Kleinsteuerungskonfigurationen auf der Basis von U 8820/U 2716 oder U 8840/U 2732 wurde deshalb ein Interpreter erarbeitet, der es gestattet, ein aufgestelltes Petri-Netz direkt (auf der Basis der Assemblerprogrammierung) in Steuerungssoftware zu übertragen. Die Assemblerprogrammie-

rung beschränkt sich auf die Beschreibung der Tansitionsbedingungen mit den dazugehörigen Vorgänger- und Nachfolgeradressen sowie die Beschreibung der auszuführenden Aktionen. Ein arbeitsfähiges Steuerprogramm entsteht durch Abspeichern des programmierten Netzes zusammen mit dem Interpreter auf EPROM.
Der Interpreter kann nachgenutzt werden. Es wird der Quelltext übergeben, um Anpassungen an die konkrete Hardware vornehmen zu können.

Konsum-Bürstenfabrik, Schönheider Straße 61, Stützengrün, 9415; Tel. Rothenkirchen/V. 34 11, App. 232 *Luf*

Abrechnung aller Fuhrparkleistungen

Das Softwareprojekt *Abrechnung aller Fuhrparkleistungen* ermöglicht die Rationalisierung der Abrechnung von Fuhrparkleistungen auf der Grundlage der GKT 370, der AO 151, der Personenbeförderungsordnung PAO 2014 sowie gültiger Betonpreise (Rechentechnik: MS-DOS-kompatibler PC; Programmiersprache: dBase III Plus, Clipper). Die gesamten Fuhrparkleistungen (Baumaschinen, Güterkraftverkehr, Leihsätze, Personenverkehr, „freie“ Rechnungslegung, Abrechnung Betonproduktion) werden getrennt nach innerbetrieblichen und Fremdkostenstellen erfaßt und abgerechnet. Es erfolgen eine Rechnungsstellung und eine Auswertung nach verschiedenen Parametern (Kraftfahrzeuge, VK, DK und innerbetriebliche Kostenstellen).
Eine individuelle Rechnungslegung kann für Rechnungen nach frei wählbaren Parametern, z. B. Kiesrechnungen, Leihsätze für Betriebsangehörige, erfolgen. Sie erfolgt durch Eingabe von eigenen Preisen, Maßeinheiten und Preisbasen. Es kann eine Auswertung nach Fahrzeugen, Baumaschinen, innerbetrieblichen Kostenstellen, Betonproduktion und Personenkraftverkehr erfolgen. Weiterhin ist eine Auswertung des DK- und VK-Verbrauches möglich.

ZBO Landbau Delitzsch, Schkeuditzer Straße 70, Delitzsch, 7270; Tel. 55 61 *Krippner*

Dateibestandskontrolle unter DCP

Auf der Grundlage eines Neuerervorschlages wurde ein Programm zur Dateibestandskontrolle entwickelt. Das Programm **FILETEST** hat die Aufgabe, Dateiänderungen auf einem Datenträger (Diskette/Festplatte) festzustellen. Es werden alle Dateien erfaßt, die sich auf dem Datenträger befinden. In Abhängigkeit von Parametern in der Kommandozeile liefert es die Veränderungen des Dateibestandes seit der letzten Archivierung. Archiviert und ausgewertet werden Informationen aus den Verzeichniseinträgen der Dateien, wie Dateiname, Dateilänge, Verzeichnisname. Optional erfolgt eine Kontrolle der Dateiintegrität der lauffähigen Programme und Treiber bzw. aller Dateien durch einen schnellen Prüfsummenalgorithmus. Generell werden Prüfsummen des Disketten-/Festplattenurladers berechnet und ausgewertet. Weiterhin können die Dateien nach dem Auftreten einer frei wählbaren Bytefolge durchsucht werden. Die Unterschiede in den archivierten Informationen zwischen zwei Aufrufen des Programms FILETEST werden auf dem Bildschirm angezeigt, in einer Textdatei abgespeichert und können optional auf Paralleldrucker ausgegeben werden. Um die Integrität des Programms FILETEST zu sichern, werden nach dem Programmstart Selbsttests im Hauptspeicher und auf dem Datenträger durchgeführt, auf dem es sich befindet. Die Selbsttests bestehen in der Berechnung und der Auswertung von Prüfsummen. Der Zugriff auf das Programm kann von einem Kodewort abhängig gemacht werden. Eine abrufbare Hilfestellung erleichtert die Arbeit mit FILETEST.
Zu diesem Programm wird eine kurze Dokumentation angeboten, die neben der Beschreibung des Programmablaufs, der Eingangsdaten und der verwendeten Dateien auch Hinweise zur Fehlerbehandlung gibt.

Hochschule für Verkehrswesen „Friedrich List“ Dresden, WB Elektrische Bahnen, PSF 103, Dresden, 8072; Tel. 4 62 24 29 *Prof. Dr. Biesenack*

PICPLOT zum Plotten von Bildfiles

Für ein weitverbreitetes Softwarepaket auf 16-Bit-Rechentechnik wurde ein Programm zum Plotten der Bilddaten ohne die zeitraubende Erzeugung eines Plotfiles realisiert. Durch geringe Anpassung werden auch Bilddateien des Grafiksystems PCCAD Version 3.04 geplottet. Das Programm PICPLOT ist für verschiedene Plottertypen verfügbar und in der Abarbeitung wesentlich schneller als die Originalprogramme der Softwarehersteller.
Die Stiftdicke der einzelnen Plotterstifte ist zwischen 0 und 2,999 mm wählbar. Eine Stift-Ebene-Zuordnung ist möglich. Es erfolgt eine Auswahl der Ebenen, die geplottet werden sollen. Es kann eine Infodatei mit Stiftdicken und Stift-Ebene-Zuordnung gelesen und geschrieben werden.
Voraussetzung an Hardware: A 7150, EC 1834 oder kompatible mit Koprozessor 8087. Eine Festplatte ist nur notwendig, wenn Symbole im Bild sind.

VEB Werkzeugmaschinenfabrik Saalfeld, Abt. EAC, Koll. Franke, Hüttenstraße 19–21, Saalfeld 6800; Tel. 8 10, App. 614/602 *Rosenbusch*

Stromlaufpläne mit PCCAD

Mit der CAD-Software PCCAD (CAD-Grundpaket-20) und PCCAD/E (Elektronik mit Post-Prozeß) des VEB Leitzentrum für Anwendungsforschung Berlin steht ein leistungsfähiges Arbeitsmittel zur Verfügung, mit dem mit 16-Bit-Rechentechnik (EC 1834, A 7150) unter anderem Stromlaufpläne für elektronische Schaltungen entworfen werden können. Dabei wird in Symboltechnik gearbeitet, die den Entwurfsprozeß verkürzt und den Übergang zum Leiterplattenlayout ermöglicht. Zur Stromlaufplangestaltung wurde eine Symbolbibliothek erarbeitet, die 400 elektronische Bauelemente umfaßt. Sie beinhaltet:
– diskrete Bauelemente (Widerstände, Kondensatoren, Transistoren, Dioden, LED, LEA u. a.)
– analoge Schaltkreise (Operationsverstärker u. a.)
– digitale Schaltkreise (Baureihen D..., DL..., DS..., 74HCT..., U..., V... einschließlich 8-Bit- und 16-Bit-Rechnerschaltkreise, Speicher, Importbauelemente u. a.)
Die Symbolbibliothek wird durch die Datenbank ergänzt, die die Symbole programmtechnisch komplettiert.
Zur Nachnutzung werden angeboten:
– Symbolbibliothek (A-Symbole, B-Symbole)
– Datenbank
– Menüauflieger für das grafische Tablett K 6405
– Zeichnungsvordrucke für Stromlaufpläne nach TGL 31031/02
– Dokumentation.
Alle Dateien liegen auf 5,25-Zoll-Diskette vor.

VEB Robotron-Meßelektronik „Otto Schön“ Dresden, BfN (WtE 39), PSF 211, Dresden, 8012 *Rädel*

Prüfprogramm PRUEF für die Datenerfassung

Das Programm **PRUEF** dient der Prüfung von Textdateien durch nochmalige Eingabe der Werte und Vergleich mit den bereits gespeicherten Daten. Sein Haupteinsatzgebiet ist die Datenerfassung. Es gewährleistet eine hohe Datensicherheit durch die Möglichkeit, Erfassungsfehler frühzeitig zu erkennen.
Das Programm PRUEF verarbeitet Textdateien (ASCII-Code) mit einer maximalen Satzlänge von 254 Byte. Ein Satz kann maximal 50 Datenworte enthalten. Wahlweise können Sätze mit festen oder mit variablen Wortlängen geprüft werden. Bei variablen Wortlängen kann ein frei wählbares Trennzeichen verwendet werden. Pro Datei darf jedoch nur eine Satzart auftreten, d. h., Anzahl, Reihenfolge und Bedeutung der Worte in den Sätzen einer zu prüfenden Datei müssen konstant sein.
Durch Parametervorgaben wird bestimmt, welche Teile (Worte) der erfaßten Datensätze einer Prüfung auf den Wortinhalt unterzogen werden. Dadurch erhält das Programm einen universellen Anwendungscharakter.
Gerätetechnik: 8- und 16-Bit-Computer
Betriebssysteme: CP/M (und kompatible), MS-DOS (und kompatible)
Programmiersprache: Turbo-Pascal

VEB Motorenwerk Cunewalde, ODV, Bielebohstraße 1, Cunewalde, 8704 *Wendschuh*

DISKETTE

Das Programm wurde auf der Grundlage eines modernen Datenbanksystems unter dem Betriebssystem DCP 1700 3.3 für den A 7150 sowie für den EC 1834 entwickelt. Es liegt in compilierter Form als EXE-Datei vor. Der Einsatz ist für Rechner mit Festplatte vorgesehen, eine Diskettenversion ist jedoch möglich. DISKETTE ermöglicht das inventursichere Erfassen, Editieren und Verwalten von Diskettenbeständen und bietet zugleich umfangreiche Recherche- und Druckmöglichkeiten. Selbsterklärende, vor Fehlbedienung geschützte Menüs ermöglichen eine sofortige Anwendbarkeit und setzen beim Nutzer keinerlei rechentechnische Kenntnisse voraus. Das Anlegen und Rückspeichern von Sicherheitskopien wird menügeführt unterstützt.

GHG Technik, Kulturwaren, Sportartikel Halle, FD WuT, Köthener Straße 30, Halle, 4060; Tel. 60 12 12/47 *Harz*

SEAP 16 Version 2.1

Der im Mai 1989 in der MP angebotene Softwareentwicklungsarbeitsplatz für 16-Bit-PCs (SEAP 16 Version 2.0) wurde unter der Voraussetzung entwickelt und konzipiert, daß der benutzte Rechner über eine Festplatte verfügt. Um jedoch auch die Nutzer zu unterstützen, die über keine Festplatte verfügen, wurde eine entsprechende Variante von SEAP 16 bereitgestellt. Die Funktion der Festplatte übernimmt dabei zum einen die bei Kaltstart eingerichtete RAM-Disk und zum anderen ein Satz von Disketten für das Laufwerk A. Durch SEAP 16 wird der Nutzer zu eventuell notwendigem Diskettenwechsel aufgefordert. Die SEAP 16-Variante ohne Festplatte hat prinzipiell die gleiche Benutzeroberfläche wie die Festplattenvariante. Generell können alle Prozeduren und Kommandos in der gleichen Weise wie in der Version 2.0 benutzt werden, ohne daß man wissen muß, auf welcher Diskette welche Komponente (System, Kommando, Prozedur) zu finden ist.
Die Disketten der Version 2.1 erhalten inhaltsbezogene Namen. Diese erscheinen in einer automatischen Auflegeanforderung, wenn sich die aufgerufene Komponente (Betriebssystem, einzelne Programme, Software, Kommandoprozeduren) nicht auf der gerade aufliegenden Diskette befindet. Wir verteilen für Sie die einzelnen Komponenten sinnvoll und speicherplatzsparend auf mehrere Disketten.
Es steht eine Vielzahl von Kommandoprozeduren bereit, mit denen Sie mit geringstem Aufwand und ohne Spezialwissen eine gewünschte Komponente aufrufen können.
Für Nutzer des A 7100, die das Betriebssystem DOS 7100 verwenden, ist SEAP 16 ebenfalls nutzbar.

VEB Datenverarbeitungszentrum Halle, Abt. FP, Block 081, Halle-Neustadt, 4090; Tel. 61 60 *Dietrich*

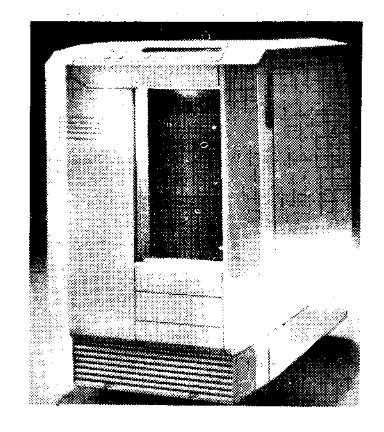
Anwendung der Tastatur TheBoard

Zu den praktischen Anwendungsmöglichkeiten der LCD-Tastatur TheBoard (s. Bilder 11 und 12 auf der 3. US) findet im I. Quartal 1990 an der Karl-Marx-Universität Leipzig eine eintägige Informationsveranstaltung statt. Interessenten wenden sich bitte mit ihrem Teilnahmewunsch bis zum 31. Januar 1990 an:

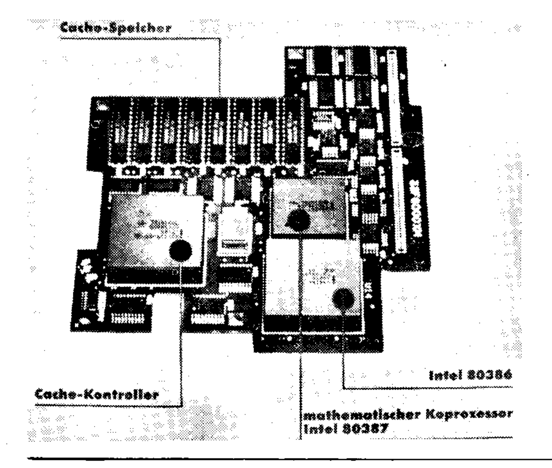
Karl-Marx-Universität Leipzig, Sektion Informatik, Labor für Softwareentwicklung, Doz. Dr. sc. Eberhard Kummerow, Karl-Marx-Platz, 10/11, Leipzig, 7010. *Dr. Kummerow*

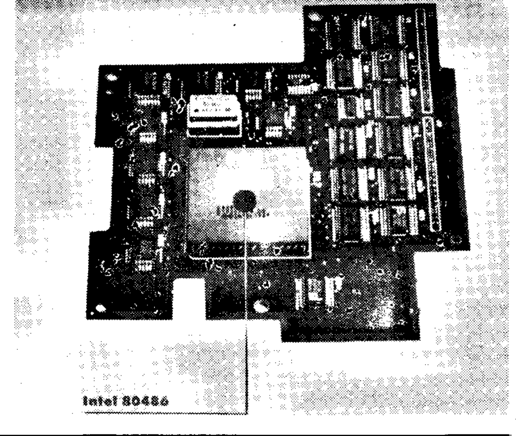
Erste 486er Applikationen

In unserem Heft 8/89, Seite 250, informierten wir Sie unter der Überschrift „Erste PC-Familie mit 80486-Prozessor" über die neue VX-Serie von Apricot. Hätte uns zu der Zeit bereits das Bild dieses Computers vorgelegen – das wir Ihnen hiermit nachreichen –, wäre die Überschrift sicherlich anders ausgefallen. Denn sowohl vom Volumen des Computers (Bild oben) wie von den Leistungsmerkmalen her läßt sich der VX eher der mittleren Datentechnik als „persönlichen" Computern zuordnen. Laut Tim Taylor, Marketingleiter von Apricot, liegt die neue Maschine demnach auch zwischen einem Personalcomputer und einem Minirechner. Die Haupteinsatzgebiete der Modelle VX FTserver 400 und 800 sind als Netzserver bzw. als Hostrechner unter Unix zu sehen.
Den Prozessor 80486 erstmals in einem Desktop-(Auftisch)-PC eingesetzt zu haben, kann IBM für sich in Anspruch nehmen. Allerdings nicht in einem eigens dafür entwickelten PC, sondern im Modell 70 des bereits bestehenden Personal Systems/2. Das Prinzip besteht darin, daß eine auf dem Motherboard befindliche Platine, die den Mikroprozessor 80386, den Koprozessor 80387 und den Cachespeicher samt Cachecontroller enthält, gegen eine Platine mit dem 80486-Prozessor ausgetauscht wird; laut IBM ein Beweis dafür, wie universell erweiterbar das PS/2 bereits angelegt wurde. Beim Kauf der neuen Platine kann die alte in Zahlung gegeben werden; den 80387 nimmt IBM allerdings nicht zurück. Die Kosten der Umtauschaktion belaufen sich auf etwa 10000 DM, als Gegenleistung wird dem Anwender eine Leistungssteigerung seines PCs von 80 Prozent versprochen. Die Bilder unten zeigen anschaulich die Integration der 386er Komponenten in den Prozessor 80486.
Inzwischen wird auch von der Firma AST für ihre PCs Premium 386 als Upgrade eine 486-Steckkarte angeboten. Das FASTboard 486/25 enthält außerdem einen Sockel für den Koprozessor Weitek 4167.



Etliche PC-Hersteller wollen solche Aufrüstungen allerdings nicht gelten lassen, da die Computer nicht speziell für den 80486 entwickelt wurden; von ihnen sind inzwischen auf der Systems 89 die ersten „echten" 80486-PCs präsentiert worden. Wir werden auch darüber in einem der folgenden Hefte berichten. *MP*





Bull-Gruppe und MIPS vereinbaren Kooperation

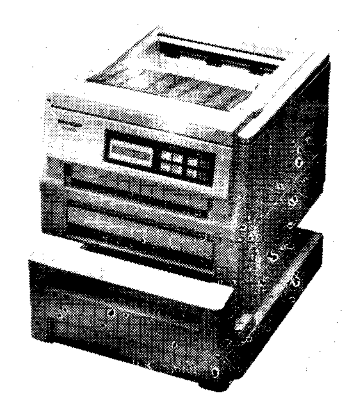
Die Bull-Gruppe, Paris, und die kalifornische MIPS Computer Systems Inc., Sunnyvale, unterzeichneten eine Vereinbarung, wonach Bull die RISC-(Reduced Instruction Set Computer-)Technologie von MIPS für den Ausbau ihres UNIX-Systemangebotes verwenden wird. Die Bull-Gruppe, die 1984 als erster europäischer Computerhersteller mit der Vermarktung von RISC-Rechnern begonnen hatte, wird die RISC-Architektur und die Software von MIPS für die Entwicklung einer neuen Systemgeneration verwenden, die das UNIX-Produktangebot nach oben erweitert. So will Bull ab April 1990 neue High-End-Systeme mit einer Prozessorleistung von über siebzig Millionen Befehlen pro Sekunde (MIPS) und mehr als 100000 Drystones/s anbieten, bei denen die MIPS 6000-Chipserie Verwendung findet. Die neuen RISC-Systeme sollen primär als leistungsfähige Datenbank-Computer und als Server zur Unterstützung von technisch-wissenschaftlichen und Büroinformationsworkstations in Netzwerken eingesetzt werden. *MP*

32-Bit-Echtzeitbetriebssystem

Von der Firma Digital Research Inc., bekannt als Entwickler des 8-Bit-Betriebssystems CP/M, wurde jetzt eine Reihe neuer Produkte angekündigt, die dem Unternehmen „einen neuen Wachstumsschub" verleihen sollen. Im 16-Bit-PC-Bereich konnte man sich ja mit dem Betriebssystem CP/M 86 gegenüber MS-DOS von Microsoft nicht durchsetzen.
Mit FlexOS 386 wird von der Firma jetzt ein Multiuser-/Multitasking-Betriebssystem speziell für die Echtzeitverarbeitung angeboten. Als Option gibt es darüber hinaus eine MS-DOS 3.3-kompatible Applikationsumgebung. FlexOS 386 läuft auf Computern mit dem 32-Bit-Mikroprozessor Intel 80386 und bietet 32-Bit-lineare und 16-Bit-segmentierte Speicheradressierung. Weitere Merkmale sind schnelle Interruptlatenzzeiten und sogenannte context-switch-Arbeitsleistungen sowie ein transparentes Netzwerkinterface. Zur Betriebssystemfamilie gehören die Systemprogramme FlexOS 186, FlexOS 286 sowie die Netzwerke FlexNet und FlexView. Als Hauptanwendungsgebiet für das neue Betriebssystem wird der Fabrikautomatisierungsbereich angesehen.
Ebenfalls vorgestellt wurde mit X/GEM eine Erweiterung der aus dem PC-Sektor bekannten Grafiksoftware GEM (Graphics Environment Manager). Erste Implementierungen dieser Entwicklung auf die Betriebssysteme FlexOS 286 und FlexOS 386 gibt es bereits. *MP*

Sharp Laserdrucker JX-9500

Besonders auf Anwendungen in den Büros, wo mit dem Platz gegeizt werden muß, zielt Sharp mit dem neuen Laserdrucker JX-9500: Mit Abmessungen von 36 mal 34 mal 26,7 cm^3 – inclusive der eingebauten Papierkassette – und reiner Frontbedienung findet er auch auf engen Schreibtischen genug Stellfläche. Dennoch werden die von anderen Tischlaserdruckern bekannten Leistungen geboten: Sechs Seiten im Format A4 druckt der JX-9500 pro Minute bei einer Auflösung von 300 dpi (das entspricht 12 Punkten pro Millimeter) wahlweise in sechs Schriftarten (Courier, Courier bold, Line Printer, jeweils Protrait und Landscape). Drei weitere Fontkarten stehen zur Verfügung, jeweils zwei davon können gleichzeitig aktiviert sein. Fünf Druckerstandards beherrscht er: HP Laser Jet II, Epson FX 80, IBM Pro Printer, IBM Graphics Printer und Diablo 630. Die Papierzufuhr kann von Hand geschehen oder aus Kassetten, die die automatische Einspeisung in verschiedenen Formaten regeln.
Der Sharp JX-9500 wird von dem Mikroprozessor Motorola 68000 gesteuert, bietet 512 KByte an RAM (aufrüstbar bis 4,5 MByte) und ist über eine Centronics- und eine RS-232C-Schnittstelle ansprechbar. Er arbeitet elektrofotografisch mittels Halbleiterlaser, die Entwicklung geschieht nach dem Zweikomponentenverfahren. *MP*



System-Software auf Compact-Disks

Während überlicherweise Software-Updates auf Magnetbändern und -platten ausgeliefert werden, setzt Hewlett-Packard alternativ Compact Disks (CD) als Trägermedium für das Betriebssystem der Computersysteme HP 3000 und über hundert MPE/V-Softwareprodukte ein. Vorteile des HP LaserRelease Service auf CD-ROM-Basis sind der einfache Vertrieb und die erleichterte Installation der Software-Updates. HP LaserRelease für die Computersysteme HP 3000 unter MPE/V enthält ein neues Verfahren zur Software-Installation, das zusammen mit einem kürzlich vorgestellten CD-ROM-Laufwerk mit HP-IB-(IEEE-488-)Schnittstelle das Laden der Software von CDs ermöglicht.
Nach Überzeugung von HP wird damit erstmals ein komplettes System von Betriebssystemen und systemnaher Software auf Compact-Disk vorbereitet. *MP*

Wärmebeständige Leiterplatten

Die Wärmebelastbarkeit von Leiterplatten wird durch die auf Epoxid- oder Acrylharz basierenden Kleber begrenzt, mit denen Kupferfolie auf dem Basismaterial befestigt ist.
Sumitomo Electric Industries entwikkelte ein flexibles Leiterplattenmaterial, bei dem Kupferfolie direkt auf der Leiterplatte befestigt wird. Die Kupferfolie löst sich auch bei höherer Einsatzdauer nicht ab. Die Leiterplatte soll neben guten elektrischen und mechanischen Eigenschaften chemische Beständigkeit und Lötfestigkeit aufweisen sowie das Anschweißen von Goldfäden gestatten.

Quelle: New Technology Japan (1989) 9 Wi

Disk-Beschriftung in 15 Sekunden

Zur schnelleren Herstellung magnetooptischer Speicherplatten entwikkelte die japanische Firma Sharp ein neues Verfahren, mit dem es möglich sein soll, die Speicherplatten in nur 15 Sekunden zu beschriften. Der bisherige Fotomasken-Laserbeschriftungsprozeß benötigte dazu eine Zeit von einer halben Stunde. Bei dem als *Contact Printing Process* bezeichneten Verfahren soll die Ausschußrate sehr gering sein, so daß Sharp damit rechnet, den neuen Prozeß als Massenfertigungsverfahren für die nächste Generation magnetooptischer Disks zu verwenden. Das Disk-Substrat wird durch Auftragen eines Fotolacks auf ein Glasscheibensubstrat geformt. Das Substrat wird im Ganzen durch eine thalliumbeschichtete Fotomaske, die die zu speichernden Informationen trägt, mit einem Ultraviolett-Lichtstrahl 15 Sekunden lang belichtet. Der belichtete Fotolack wird dann durch Entwicklung entfernt und mit Fluorkarbongas trocken geätzt. Der restliche Fotolack wird mit Sauerstoff entfernt. In einer Schlußbehandlung wird die Disk dann fertiggestellt. Die auf diese Weise hergestellten Disks sollen einen Rauschabstand von 52,7 Dezibel bei einer Aufzeichnungslänge von 0,76 Mikrometer je Bit haben. Auch bei einer Aufzeichnungslänge von 0,57 Mikrometer je Bit soll man noch einen Rauschabstand von 45 Dezibel erhalten. Damit würde sich die Disk für hochdichte Datenspeicherungen eignen.

Quelle: Blick durch die Wirtschaft vom 5. 7. 1989 Fa

Trends bei Stromversorgungen

Stromversorgungen sind bei elektronischen Geräten zu einem entscheidenden System- und Kostenfaktor geworden. Preis und Qualität besitzen einen hohen Stellenwert. In diesem Zusammenhang ergibt sich die Frage nach der Rolle von Standard-Stromversorgungen. Obwohl davon ausgegangen werden kann, daß Standard-Stromversorgungen in hohen Stückzahlen realisiert werden können, ist die Fertigung kundenspezifischer Stromversorgungen so gut entwickelt, daß eine für den Anwendungsfall optimierte Stromversorgung höhere Kostenvorteile erbringt als eine unifizierte Produktion. Gegen Standard-Stromversorgungen sprechen folgende Gesichtspunkte:
– Standard-Stromversorgungen realisieren oftmals die geforderten Parameter, sie sind technologisch jedoch nicht auf dem möglichen Niveau.
– Die Standards zwingen den Systemhersteller, sich den Dimensionen der Stromversorgung anzupassen. Somit wird ein optimales Erzeugnis verhindert.
– Stromversorgungen übernehmen immer mehr Systemfunktionen, was bei Standards nicht möglich ist.
– Standard-Stromversorgungen können den hohen Innovationsgeschwindigkeiten der Geräteentwicklung nicht folgen.
In Westeuropa werden zu über 80 % spezifizierte Stromversorgungen verwendet. Standard-Stromversorgungen kommen vorwiegend bei kleinen Stückzahlen zum Einsatz, da sie flexibel und schnell verfügbar sind.

Quelle: Elektronik. – München 38 (1989) 14. – S. 12, 14 Wie

Digitaler Großflächenfarbkopierer

Die Firma Canon entwickelte einen Farbkopierer für großformatige Kopien. Das unter der Bezeichnung *Canon Color Bubble-Jet Copier A1* angebotene Gerät kopiert Vorlagen, die in ihrer Größe zwischen einer Visitenkarte und dem Format A1 variabel sein können. Der Zoom-Faktor (von 50 bis 1 200 Prozent) ermöglicht Kopien bis zu einer Größe von $7 \times 10\,m^2$.
Nach Einschätzung der Firma ist das Gerät in der Lage, hochwertige, auch editierte Kopien von Papiervorlagen, Kleinbild- und Großformat-Negativen und Diapositiven anzufertigen. Das Gerät arbeitet mit einer Auflösung von 400 Punkten pro Zoll und mit mehr als 64 Helligkeitsstufen. Die verschiedenen Farben werden im nassen Zustand auf das Papier gebracht, können sich demzufolge auf dem Papier vermischen und ergeben flächendeckende Mischfarben.
Der Color Bubble-Jet druckt Einzelblätter aus der Kassette oder aus Rollen und Transparentfolie, wobei Overheadfolien auf der Vorderseite und Folien für beleuchtete Plakate auf der Rückseite bedruckt werden.
Der Kopierer besteht aus einem Color-Scanner, einem Color-Drucker und der Elektronik, die umfangreiche Editierfunktionen ermöglicht. Die Farben können im Original wiedergegeben oder durch Veränderungen der Intensität und durch Übersetzung in andere Farben umgewandelt werden.
Das Gerät bietet die Möglichkeit, aus zwei Vorlagen, die sich gleichzeitig auf dem Vorlagenglas befinden müssen, eine gemeinsame Kopie anzufertigen, wobei ein Teil einer Vorlage in einen Teil der anderen kopiert wird. Außerdem läßt sich mit dem Image Repeat Mode beipielsweise eine Visitenkarte blattfüllend abbilden. Dabei können einzelne Elemente eines Bildes entweder nicht oder nur an bestimmten Stellen der Kopie mehrfach gedruckt werden. Es können auch von Negativen, Diapositiven oder Transparentfolien spiegelverkehrte, positive oder negative Kopien erzeugt werden. Filme (bis zum Format $20 \times 25\,cm^2$) und Overheadfolien können direkt auf das Vorlagenglas gelegt werden. Fa.

Weitere Leistungserhöhung durch 96-Bit-Architektur

Nach Aussagen der Beratungsfirma International Technology Group soll die Firma IBM im Zeitraum von 1989 bis 1996 Forschungs- und Entwicklungsvorhaben realisieren, die auf Computer in 96-Bit-Architektur zielen und mit denen Speicher von bis zu 281 Terabyte direkt adressiert werden können. Diese Architektur soll gegenüber gegenwärtigen Rechnern eine Leistungssteigerung um den Faktor 300 bewirken.
Die Firma IBM soll sich auch mit der Kombination funktionell spezialisierter Prozessoren, der Realisierung von Systemarchitekturen auf Chip-Ebene, der RISC-Architektur und optischen Datenübertragungsverfahren beschäftigen.

Quelle: Electronic Design. – New York 37 (1989) 10. – S. 23 Wie

Fernsehbilder auf dem PC

Auf dem Gebiet der Verarbeitung von Fernsehbildern durch Computer arbeitet ein Entwicklerteam der britischen Firma Videologic. Es entwikkelte eine Leiterplatte mit der Bezeichnung *Digital Video Adapter (DVA) 4000* und eine spezielle Mikrochip-Reiche. Wird dieser DVA an einen PC angeschlossen, so wird die Verarbeitung von Videobildern möglich, was bedeutet, daß ein Fernsehbild auf einem herkömmlichen Computerbildschirm dargestellt werden kann. Die Bilder können nicht nur vergrößert, verkleinert, als Bildschirm-„Fenster" angezeigt und gespeichert werden, sondern auch ausgeblendet, gemischt und als Festbild dargestellt und überlagert werden. Dabei folgt das Videobild der „Grammatik" des Fernsehens. Mit dieser DVA wurden neue Verbindungen zwischen den modernen Fernsehbildern und der für das heutige Video typischen groben grafischen Darstellung und scharfen Überblendung von Videobildern geschaffen. Außerdem kann ein über Funk oder von einer Laserdisk übernommenes Fernsehbild in Binärziffern umgewandelt werden. Das digitalisierte Bild kann anschließend auf Computerbildschirm in Farbe und mit hoher Bildschärfe dargestellt werden. Ist das Signal erst einmal digitalisiert, so kann es durch den Computer auf verschiedenste Art verarbeitet werden. Das Problem bei interaktiven Videosystemen liegt in der geringen Menge an Informationen, die in einem herkömmlichen analogen Fernsehsignal enthalten sind. Wenn man den DVA 4000 an einen Rechner anschließt, kann er zur Verarbeitung bewegter Bilder 25 Millionen Bit an Informationen/Sekunde abarbeiten. Im Normalbetrieb würde ein Rechner in einer Sekunde nur eine Million Bit an Informationen verarbeiten. Der größte Chip von Videologic ist von ähnlicher Komplexität wie ein Motorola 68000. Er umfaßt einen kompletten Fernsehtuner, das heißt, ausgestattet mit der DVA 4000 erweitert sich ein PC zu einem Fernsehgerät mit hoher Bildschärfe.

Quelle: The Wall Street Journal vom 7. 7. 1989 Fa

Nitridfilm für Magnetköpfe

Größere Aufzeichnungsfrequenzen und höhere Speicherdichten erfordern bei größeren Koerzitivkräften neue Materialien für Magnetköpfe. Der Firma Matsushita Electric Industrial Co., Ltd. gelang die Entwicklung eines neuen Nitridlegierungsfilmes für Magnetköpfe. Dieser Film weist eine Sättigungsfeldstärke von 13 bis 15 KG auf und hat eine thermische Stabilität, die eine Wärmebehandlung bis 700°C erlaubt. Bisher gab es noch kein Material, das hohe magnetische Sättigung mit entsprechender thermischer Stabilität vereinigt. Der neue Film wird durch zyklisches Zumischen von Stickstoff zu einer Legierung erzeugt, die aus Niobium, Tantal und Zirkonium besteht. Hergestellt wird der Film durch Sputtern. Er besteht durch das spezielle Verfahren abwechselnd aus Nitridschichten und solchen ohne Stickstoffanteil. Im Ergebnis dessen konnten die magnetische Sättigung um 30 bis 50 Prozent und die magnetische Anfangspermeabilität bei 1 MHz um einen Faktor von rund 3 000 vergrößert werden. Durch die hohe Wärmebeständigkeit ist eine Verbindung mit Glas möglich, und es lassen sich Magnetköpfe mit Spaltbreiten von unter 0,2 nm herstellen.

Quelle: New Technology Japan vom Mai 1989 Fa

Gemeinschaftsentwicklung einer Arbeitsstation

Unterstützt durch die Europäische Gemeinschaft wollen mehrere westeuropäische Unternehmen und Universitäten im Rahmen des Esprit-Programms gemeinsam einen Hochleistungscomputer entwickeln, der als Arbeitsstation konzipiert ist. Bis Mitte der neunziger Jahre soll eine Arbeitsstation entstehen, die die Rechnerleistung heutiger Supercomputer aufweist (1 000 Megaflops).
Nach dem gegenwärtigen Konzept ist eine Rechnerarchitektur mit einem geclusterten Multiprozessorsystem vorgesehen, das über einen Hochgeschwindigkeitsbus mit Cachekonsistenz, mehrere lokale und einen zentralen Speicher verfügt. Neben Standardprozessoren soll ein schneller KI-Prozessor zum Einsatz kommen. Ein 3-D-Subsystem wird hochauflösende Grafik möglich machen.
Das Standard-Unix-Betriebssystem soll um Multiprozessor- und Multicomputeranwendungen erweitert werden. KI-Anwendungen werden durch die Sprache Prolog unterstützt.

Quelle: Blick durch die Wirtschaft vom 9. 8. 1989 Wie

Digitale Automaten

von M. Krapp, VEB Verlag Technik, Berlin 1988, 244 Seiten, 153 Bilder, DDR 30,– M

Das vorliegende Fachbuch stellt eine anspruchsvolle, theoretisch gut fundierte und doch verständliche Einführung in das gesamte Wissensgebiet digitaler Automaten dar.
Ein mathematischer Grundlagenabschnitt führt in die Mengen- und Schaltalgebra ein. Darauf aufbauend werden kombinatorische Schaltungen, besonders ihre Dekomposition und Minimierung, behandelt. Der nächste Abschnitt ist dem Entwurf von sequentiellen Automaten auf der Grundlage der Modelle von Moore- und Mealyautomaten gewidment. Ein weiterer Abschnitt beschäftigt sich mit Fragen der Dekomposition und Komposition komplexerer Systeme digitaler Automaten. Der letzte große Abschnitt behandelt den Computer als programmierbare digitale Maschine. Neben der Darstellung des weit verbreiteten klassischen Von-Neumann-Konzepts werden auch Ausblicke auf Transputer und Datenflußarchitekturen gegeben.
Auffallend ist die Darlegung aller Abschnitte auf der didaktisch-methodischen Grundlage der Theorie digitaler Automaten, zu deren Verständnis vor allem das umfassende Symbolverzeichnis und der mathematische Grundlagenabschnitt beitragen.
Weiterhin zeichnet sich das Fachbuch durch eine wissenschaftliche Strenge aus. Jeder benutzte Fachbegriff wird definiert, was vor allem deshalb erwähnenswert ist, weil noch nicht alle Fachbegriffe eine feststehende unumstrittene Bedeutung haben. Damit wird dieses Fachbuch auch für viele Praktiker von hohem Wert sein.
Hervorzuheben sind weiterhin die vielen, mit Bedacht ausgewählten, praxisrelevanten Beispiele, die sich vor allem an den technisch interessierten Leser wenden.
Auf der Grundlage der genannten positiven Eigenschaften könnte das Fachbuch von Krapp auf dem Teilgebiet des Entwurfs digitaler Systeme in Lehre und Praxis zu einem Standardwerk werden.

Prof. Dr. Th. Horn

DOS kompakt und komplett

von K. Jamsa, Übersetzung und Überarbeitung: M. Däumeling, McGraw-Hill Book Company GmbH, Hamburg 1988, ISBN 3-89028-179-9

Eine vollständige Übersicht über die internen und externen MS-DOS-Kommandos und Befehle – einschließlich der Version 4.0 – auf nur 130 Taschenbuchseiten bieten zu wollen, ist ein Vorhaben, das sicher von vielen Programmierern begrüßt wird, und das vorliegende Ergebnis kann als gelungen angesehen werden. Nach einem kurzen Vereinbarungsteil mit den im Buch verwendeten Konventionen wird sofort begonnen, das im Titel gegebene Versprechen einzulösen. Um es gleich vorwegzunehmen: Komplett ist es, und ob ein Taschenbuch auch kompakt ist, mag jeder für sich entscheiden. Komplett vor allem auch deshalb, weil dieses kleine Werk nicht nur die zur direkten Arbeit mit den Dateien und zur Nutzung der zahlreichen Funktionen von MS-DOS erforderlichen Befehle enthält, sondern auch Hinweise zur Konfiguration des Betriebssystems mit Hilfe der Datei CONFIG-.SYS. Dies gibt insbesondere dem Anfänger die Möglichkeit, die Standardeinstellungen von DOS seinen individuellen, auch landestypischen, Erfordernissen anzupassen, was ja bereits vor dem Start des ersten Anwenderprogramms dessen Bedienkomfort und effektiven Lauf mit beeinflussen kann. In der nachfolgenden Darstellung sind zu jedem DOS-Kommando neben dessen Format und Syntax die möglichen Parameter aufgeführt und ausführlich erläutert. Auf Besonderheiten der einzelnen Kommandos wird hingewiesen und die Anwendung an leicht nachvollziehbaren Beispielen verdeutlicht. Die Wirkung der Parameter kann ebenfalls an Beispielen geübt werden. Damit gibt das Buch vor allem dem Ungeübten, von komfortableren Bedienoberflächen Verwöhnten die Möglichkeit, sich aus der Vielzahl der Befehle und Parameter das für seinen Zweck Passende herauszusuchen und auszuprobieren.
Alle im Buch erläuterten Befehle, Kommandos und die jeweils möglichen Parameter beziehen sich, wenn nichts anderes angegeben ist, auf die Version 2.0. So kann jeder leicht feststellen, welche konkreten Möglichkeiten seine spezielle Version bietet.
Eine angenehme Bereicherung sind die Befehle für die Stapelverarbeitung in Batch-Dateien. Sie sind auch als solche beschrieben, aber leider unter die reinen DOS-Befehle gestreut. Ein gesonderter Abschnitt, beipielsweise nach der Systemkonfiguration, mit einem Hinweis auf die Anwendung in der Datei AUTOEX-EC.BAT wäre sicher wünschenswert. Alles in allem kann dieses kleine, handliche Büchlein ein große Hilfe in der täglichen Praxis sein.

J. Hill

Wörterbuch der Datenkommunikation

Englisch–Deutsch/Deutsch–Englisch

Von Walter Tietz, 2., völlig neubearbeitete und erweiterte Auflage, Heidelberg: R. v. Decker's Verlag, G. Schenck 1989, 297 S., kartoniert, 48,– DM, ISBN 3-7685-1088-3

In der Reihe Taschenbuch Telekommunikation – in der unter anderem bereits Bändchen zu ISDN und DATEX erschienen sind – kam in diesem Jahr die zweite Auflage des zweisprachigen Wörterbuches der Datenkommunikation heraus. Neben den jeweils etwa 4 300 Stichwörtern in Englisch und in Deutsch (über 1 000 mehr als in der 1. Auflage) enthält es einen sehr wertvollen Anhang mit mehr als 1 500 Abkürzungen von Wörtern und Organisationsnamen, die im Zusammenhang mit der Datenkommunikation stehen. Für dieses eng abgegrenzte spezielle Gebiet der Elektronik stellt das Wörterbuch ein praktisch lückenloses Verzeichnis der zur Zeit gültigen Begriffe dar.
Da in vielen Fachtexten – oft aus Verlegenheit – englische Fachbegriffe nicht übersetzt werden, können solche Wörterbücher zur Verbesserung der Verständlichkeit von Publikationen in Deutsch beitragen. Jede Art von Wörterbuch – also sowohl Duden wie auch Fachwörterbuch – wird jedoch von vielen auch als Regelwerk benutzt. Deshalb sollte bei der Benutzung des Wörterbuches der Datenkommunikation in der DDR beachtet werden, daß einige Wörter in ihrer Übersetzung oder ihrer Rechtschreibung nicht unserem Sprachgebrauch bzw. unserem Duden entsprechen. Hierfür einige Beispiele, die mir am wichtigsten erscheinen:
Bei der Übersetzung des englischen Wortes *syntax* gibt der Autor den Begriff Sprachaufbau an, syntax bedeutet aber Satzbau. In der Informatik wird für syntax entweder Befehlsaufbau oder Befehlsstruktur verwendet.
Das Wort *physical* hat im Deutschen zwei Bedeutungen: physisch (zum Beispiel im Unterschied zu logisch) und physikalisch (im Unterschied zu chemisch oder biologisch). W. Tietz findet für physical zwar auch die Wörter körperlich und brieflich, jedoch statt physisch gebraucht er jedesmal physikalisch. Neben der „logischen Verbindung" fand ich in seinem Wörterbuch folglich nur die „physikalische Verbindung".
Vorsicht ist auch geboten bei der Verwendung von zusammengesetzten Wörtern, insbesondere bei ihrer Groß-/Kleinschreibung (beispielsweise Ein-Kanal-pro-Träger oder Nicht-erreichbar). Der Bindestrich wird viel zu häufig angewendet. Dadurch sind Sinnzusammenhänge (von durch Bindestrich getrennten Wörtern) nicht mehr erkennbar. Sie werden der angestrebten Übersichtlichkeit geopfert (zum Beispiel Benutzer-Speicher-Transfer-Station). Das Fugen-s koppelte W. Tietz meistens mit dem Bindestrich (beispielsweise Registrierungs-/Löschungs-Anforderungs-Rückweisungsblock, aber richtig: Netzkennungsanforderung-Unterscheidungskennzeichen), obwohl in der deutschen Sprache nur *ein* Fugenelement (bei diesen Beispielen Bindestrich *oder* Fugen-s) zulässig ist.
Durch die konsequente Arbeit mit Querverweisen und die Hinweise, aus welchem Fachgebiet die Begriffe stammen, – nicht zuletzt auch durch das handliche Format – kann das Wörterbuch der Datenkommunikation allen Nachrichtentechnikern und allen Informatikern, die sich mit der Rechnervernetzung befassen, ein wertvolles Hilfsmittel beim Lesen und besonders beim Schreiben von Fachtexten in Englisch werden.

H. Hemke

Schriftenreihe Beiträge zur Arbeitsumweltgestaltung

Im Rahmen der Schriftenreihe des Amtes für industrielle Formgestaltung erschienen jetzt zwei weitere Titel.
Das Heft 8 „Ergonomie – Leitfaden und Daten für Designer und Projektanten" (43,– M) stellt eine Art „Handlungsanleitung" dar, in der immer umfangreicher und komplexer werdendes Spezialwissen in einer auch für Nichtspezialisten anwendbaren Form dargeboten wird. Die Broschüre enthält das nach Auffassung der Autoren Dr. Sorg und Dr. Wegner notwendige Maß an ergonomischem Fachwissen zur Dimensionierung von Arbeitsplätzen und Arbeitsmitteln für Designer, Architekten, Konstrukteure, Arbeitswissenschaftler und Projektanten. Statt der sonst allgemein üblichen umfangreichen und zum Teil widersprüchlichen Tabellenmaterialien erleichtern die beigegebenen Tabellenschieber und Datenblätter für Körpermaße, Aktionskräfte und Drehmomente sowie zur maßlichen Gestaltung von Arbeitsplätzen die Arbeit mit ergonomischem Faktenmaterial.
Im Heft 9 (28,– M) mit dem Titel „Modellprojektierung und Modellsimulation im Arbeitsumweltdesign" werden für traditionelle zwei- und dreidimensionale Modelltechniken neue Anwendungsgebiete vorgestellt. Der Autor Dr. Uhlmann wendet sich mit neuen Ergebnissen praxisorientierter Forschung an Projektanten und alle auf dem Gebiet des Entwerfens und Gestaltens von Arbeitsstätten Tätige. Der als Leitfaden konzipierte Arbeitsbericht stellt in anschaulicher Art und Weise Entwurfsmethodiken sowie Hilfs- und Veranschaulichungsmittel vor, die geeignet sind, die Komplexität und Ganzheitlichkeit des Objektbereiches Arbeitsumwelt aufzunehmen und widerzuspiegeln. Dabei wird nach dem vom Verfasser entwickelten Konzept des Einheitsprojektes, d. h. der Einheit von Objekt- und Tätigkeitsgestaltung, vorgegangen. Mit verschiedenen Modelltechniken, der Modellsimulation als Vorwegnahme noch zu schaffender Realität und Problemlösestrategie auf der Basis von Gruppenkonzepten unter Einbeziehung der späteren Nutzer, wird ein praktikables, geschlossenes Methodeninventar für Arbeitsumweltdesignaufgaben erläutert.
Bereits 1988 erschienen das Heft 6 „Arbeitsumweltdesign – Seminar im Bauhaus Dessau" (15,– M) mit Beiträgen von Praxispartnern zur Vorbereitung und Realisierung komplex gestalteter Arbeitsumwelt sowie das Heft 7 „Automatisierte Fertigung – Arbeitsgestaltung und Design" (27,– M) mit Beiträgen zur Tätigkeitsgestaltung in Leitständen, zur Softwareergonomie, zu den räumlich-gegenständlichen Bedingungen und zu Designvorstellungen für CIM-Konzepte.
Die Hefte können bestellt werden beim Ministerrat der DDR, Amt für industrielle Formgestaltung, Abt. Arbeitsumwelt, Semperstr. 15, Dresden, 8020. Eine Abonnementbestellung der Schriftenreihe ist möglich.

Leipziger Herbstmesse 1989

Technikangebot von Leitthema „Flexible Automatisierung“ geprägt

Zur Leipziger Herbstmesse 1989, die vom 3. bis 9. September stattfand, kamen wieder rund 6000 Aussteller, darunter 2800 Kombinate, Außenhandels- und Exportbetriebe aus der DDR und 1400 Unternehmen aus allen kapitalistischen Industrieländern Westeuropas, aus Japan, den USA, Kanada, Australien und aus Berlin (West). Innerhalb der neun Technik-Branchenkomplexe der Herbstmesse verdienen die Branchen Textilmaschinen und Polygraphische Maschinen besondere Aufmerksamkeit. In- und ausländische Aussteller hatten sich dem Leitthema der Leipziger Messen 1989 „Flexible Automatisierung“ gewidmet. Sie demonstrierten auf vielfältige Weise, wie mit Hilfe solcher Schlüsseltechnologien wie der Mikroelektronik, der Robotertechnik, der CAD/CAM-Technik und mit flexiblen Fertigungssystemen Schritt für Schritt Grundlagen für den Übergang zum automatisierten Betrieb der Zukunft geschaffen werden. Aus dem Angebot haben wir für Sie Hard- und Softwareprodukte zum Leitthema und weitere interessante Neuentwicklungen ausgewählt.

Der VEB Aprotex Limbach-Oberfrohna und der VEB Kombinat TEXTIMA stellten gemeinsam eine neue Technologie für die Fertigung von Trikotagen vor (siehe Farbbild 1; alle Farbbilder befinden sich auf den Umschlagseiten 2 und 3). Diese teilautomatisierte Lösung wird erstmalig in dem 1988 gegründeten VEB Aprotex angewendet. Aprotex steht für **automatisierte Produktion textiler Erzeugnisse**. TEXTIMA-Großrundstrickmaschinen mit modernsten Zusatzeinrichtungen, Veredlung nach dem Farbmetriksystem und mit mikrorechnergesteuerter Färbemaschine, Stanztechnik im Zuschnitt, automatische Hängetransportanlagen und eine rechnergesteuerte Produktion bilden die Voraussetzung für eine teilautomatisierte Fertigung von Trikotagen, wobei einige Abschnitte vollautomatisiert werden. Der VEB Kombinat TEXTIMA ist wesentlich an der Ausrüstung dieses modernen Trikotagenbetriebes beteiligt. Die Technologie, die auf dem Messestand am Beispiel der T-Shirt-Fertigung demonstriert wurde, läßt sich auf die Fertigung weiterer Trikotagen übertragen. Sie ist besonders rentabel für die Stapelproduktion von Unterwäsche und Freizeitbekleidung. Ein Betriebsdatenerfassungssystem mit 240 Datenterminals registriert kontinuierlich wichtige Daten von allen Maschinen – von der Strickerei bis zur letzten Nähmaschine. Über ein lokales Netz verbundene Computer steuern den gesamten Produktionsprozeß einschließlich Transport und Lagerung. Ein umfangreiches Softwarepaket gestattet die rechnergestützte Planung und Steuerung und liefert Entscheidungshilfen für Flexibilität und Auslastung der Fertigungsanlagen.

Unter dem Motto der diesjährigen Messe stand auch das Softwareangebot des CAD/CAM-Zentrums des VEB Polygraph Leipzig. Eine durchgängige CAD/CAM-Lösung ermöglicht die Konstruktion eines Teiles, die NC-Programmerstellung und auch die Fertigungsüberwachung auf PC-Technik (siehe auch Farbbild 2). **PolyCAD** wurde mit Turbo-Pascal 5.0 entwickelt und kann auf 16-Bit-PCs mit MS-DOS 3.20 (IBM-PC und kompatible, A 7150, EC 1834) eingesetzt werden. Neben einem modularen Aufbau zeichnet sich PolyCAD insbesondere durch Schnittstellen zu internationalen Systemen wie AUTOCAD, CADdy, MEDUSA sowie zum GKS aus. Hinzu kommen die Einbindung von UNIGRAPH sowie die Nutzung beliebiger peripherer Technik; eine individuelle Anpassung an spezielle Geräte ist ebenfalls möglich. PolyCAD ist netzfähig und setzt sich aus folgenden Modulen zusammen:
Das 2-D-Konstruktionspaket PolyCON beinhaltet neben integrierten Datenbankfunktionen einen umfangreichen Komplex grafischer Prozeduren, die der Bemaßung, der Arbeit mit Symbolen und in Ebenen ebenso wie als Kommandosprache dienen.
Für den Entwurf von Projektierungssystemen mit alphanumerischen und grafischen Ergebnisdarstellungen wurde das relationale Datenbanksystem PolyDAT entwickelt. Vom Anwender wird das Datenformat ausgewählt und eingestellt, so daß vorhandene Dateien ohne Konvertierung genutzt werden können. Das Bearbeiten von Binärdateien aus Programmen in höheren Programmiersprachen ist ebenso möglich, wie auch eigendefinierte Formate, beispielsweise von Bilddateien, zu lesen und zu schreiben.
Sollen die Schnittstellen zu anderen CAD-Systemen genutzt werden, wird das Konvertierungspaket PolyKONV zu einem wichtigen Hilfsmittel bei der Übertragung der Zeichendateien von und zu PolyCON.
Für das Erzeugen von Quellprogrammen zur Steuerung der NC-Maschinen kann PolyNC genutzt werden. Ein syntaxgesteuerter Editor erleichtert dabei die Arbeit wesentlich.
PolyCAD ist somit keineswegs nur auf den Bereich der Polygraphie beschränkt, wie der Name vermuten läßt, sondern wird den Anforderungen unterschiedlichster Industriebereiche gerecht und kann insbesondere im Maschinenbau und in der Elektrotechnik eingesetzt werden.

Eine spezielle Richtung des CAD ist das Herstellen von Druckvorlagen mit Hilfe von PCs. Traditionelle Hersteller von Satztechnik setzen dafür verstärkt MS-DOS-kompatible Technik ein; dadurch ermöglichen sie den Einsatz ihrer Technik auch für Desktop Publishing – für die Text- und Bilderfassung, den Umbruch und die Seitengestaltung am Schreibtisch. Ein Beispiel dafür lieferte die Eschborner Firma Linotype (BRD) zur LHM '88 mit der Lichtsatz-Serie 2000 (wir berichteten in MP 12/1988 darüber). In diesem Jahr stellte Linotype erstmals in Leipzig den **Page Manager 410** (Farbbild 3) vor. Dieses System dient dem Textumbruch, der Layout-Entwicklung und der Ganzseitengestaltung. Es ist sehr leistungsfähig in Verbindung mit einem Zentralrechner der Linotype-Systeme 2/4/6/8, kann aber auch selbständig arbeiten. Eine enorme Gestaltungshilfe ist das grafische Tablett. Es ist mit zwei unterschiedlichen Kommandotafeln belegt – eine für redaktionelle Anwendungen und eine spezielle für die Anzeigengestaltung. Per Maus wird die entsprechende Funktion angefahren und mit einem „Klick“ aktiviert. Der geübte Bediener kann aber die Funktionen des grafischen Tabletts auch direkt über die Tastatur ansteuern. Bei den Gestaltungsvariationen sind alle Möglichkeiten vorhanden. Das beginnt beim Negativsatz und reicht bis zum rotierten Text. Konturensatz mit automatisch einfließendem Text, Definition von Rasterflächen, Kombination von Rasterflächen und Text, Umrandungen vielfältigster Art, Zeichnen von Polygonen und Kreisen – das alles läßt sich ohne großen Kommandoaufwand durchführen. Die Hardware basiert weitgehend auf PC-Technik. Die Konfiguration besteht aus einem hochauflösenden Bildschirm (48 cm diagonal) und einem zusätzlichen Standard-Monitor für die reine Textverarbeitung. Dazu gehören weiterhin eine Spezialtastatur mit Maus und der Rechner mit dem Prozessor 80286 im Tower. Das Betriebssystem ist MS-DOS; die Speichergröße ist 2,5 MByte, die Festplattengröße 40 MByte.

Zur Ausgabe der fertigen Seiten auf einem Laserdrucker (z. B. für Korrekturabzüge) oder auf einem Laserbelichter (z. B. für Druckfolien) müssen die Daten in das dafür bewährte PostScript-Format umgerechnet werden. Das übernimmt der Raster Image Processor (RIP), von dem Linotype bereits die 3. Generation vorstellte. Er wird als Tower oder integriert in den Laserbelichter angeboten. Im Farbbild 4 sind die Towerversion des **RIP 3** und der **Linotype Laser Printer 8/4 SX** zu sehen. Die neue RIP-Version zeichnet sich durch beachtliche Erweiterungen der Speicherkapazität aus: 8-MByte-Arbeitsspeicher, 135-MByte-Festplatte. Ebenso wurde die Taktfrequenz des MC 68020 von 16,7 auf 25 MHz gesteigert. Darüber hinaus bietet die Stand-alone-Version noch eine weitere Einsatzmöglichkeit: das Einbinden in ein Ethernet-kompatibles LAN. Und zum Lieferumfang des RIP 3 gehören jetzt 35 Schriftfonts. Das heißt, der RIP gibt nicht irgendeine Schrift schlechthin aus. Die Schriften von Linotype, die unter anderem so bekannte Markennamen wie Times, Helvetica (die Schrift der MP) und Optima enthalten, sind in der grafischen Industrie schon lange zum Standard geworden. Darüber hinaus haben aber auch für den DTP-Bereich bereits über 80 Firmen Lizenzen an den Schriften von Linotype erworben – darunter so bekannte Namen wie Adobe (der PostScript-Erfinder), Apple, DEC, IBM, Matsushita, NEC, Sun und Xerox. Bekanntlich lassen sich alle PostScript-Schriften von Linotype problemlos auf PostScript-kompatible Laserbelichter und Laserdrucker zuladen und sind sowohl auf dem Apple-Macintosh als auch auf IBM-PCs einsetzbar. Die unter der Bezeichnung Linotype Library angebotenen Schriften im Original-PostScript-Format umfassen zur Zeit rund 400 unterschied-

Computer Analysis of Images and Patterns

Das Thema der diesjährigen Messekonferenz, der CAIP '89, war die automatische Bildverarbeitung. Neben der Kammer der Technik als Veranstalter beteiligten sich auch die Akademie der Wissenschaften der DDR, die Friedrich-Schiller-Universität Jena und die Kombinate Carl Zeiss JENA und Robotron an der Ausrichtung der Konferenz. Die weitere Automatisierung der Produktion erfordert beispielsweise auch eine automatische Erkennung der zu montierenden Einzelteile und die automatische Überwachung der Produktion und deren Qualität. In vielen Vorträgen wurde, ausgehend von der Theorie der Bildverarbeitung, so beispielsweise zur digitalen Verarbeitung von dynamischen Bildern, zur Modellierung von 3-D-Objekten von Fernsehbildfolgen oder zur Bestimmung der Farbanteile beim Mehrfarbendruck, auch die Hard- und Software behandelt. Das interaktive Bildverarbeitungs- und Anzeigesystem ATLAS-90 und Kodierungstechniken für die HDTV-Übertragung wurden ebenso vorgestellt wie ein Expertensystem für die Anwendung von Unterprogrammen zur Bildverarbeitung, die Möglichkeiten der Verbindung der Rasterbildanalyse mit der Computergrafik und die Übertragbarkeit von Bildverarbeitungssoftware. Eine Posterausstellung zu den Vorträgen bereicherte die Konferenz und ermöglichte die weitere Diskussion in den Konferenzpausen.

liche Garnituren und stellen damit nach Linotype die größte PostScript-Fontbibliothek dar.

Aber auch die Firma Monotype (Frankfurt/Main) gehört zu den traditionellen Entwicklern von Schriften (z.B. Garamond und moderne Antiqua). Seit 1979 in Leipzig nicht mehr vertreten, war Monotype dieses Mal mit dem Publishers Electronic Network-System – **PENS** – präsent. Diese von der britischen Firma Miles 33 Ltd. erstmals 1988 vorgestellte Software wurde von Monotype ins Deutsche übersetzt und auf die PCs des Monotype-Partners Siemens gebracht. Ein Zentralrechner wird nicht benötigt. Auch das Ethernet-kompatible SK-Net Junior, über das die MS-DOS-Rechner miteinander kommunizieren, wird nicht von Monotype hergestellt. Dieses 10 MBit/s schnelle Ethernet stammt von Schneider & Koch; die LAN-Software ist von Novell. Alle Programme und Daten werden in einem Fileserver (SICOMP PC 32-20) abgelegt, das Rechnen übernehmen die einzelnen Arbeitsplätze sowie die Ausgabesteuerung.
Monotype sieht die Stärken von PENS im kleineren und mittleren Zeitungsbetrieb. Der Nutzer ist bei der Texterfassung mit seinem PC mit EGA-Farbbildschirm und spezieller Miles 33-Tastatur in der Lage, Schriftarten in verschiedenen Farben kenntlich zu machen (siehe Farbbild 5). Die Satzsteuerzeichen und Kombinationen davon können mit dem Druck auf eine der speziellen Funktionstasten in den Text eingebaut werden. Am Arbeitsplatz für den Umbruch (Farbbild 6) werden die Flächen für Bilder reserviert und numeriert und der Text eingespiegelt. Dabei sind verschiedene Funktionen möglich: Linienrahmen, Spaltenlinien (die in der Höhe mitlaufen), vertikaler Keil, Negativschrift, rotierte Zeilen, verschiedene Rasterwerte und -winkelungen sowie mehrere Arten von Ecken. Zusätzlich kann ein Ausnahmelexikon geführt werden.
Der Umbruch-Arbeitsplatz arbeitet mit 3 MByte RAM, 40-MByte-Festplatte, Taktfrequenz 16 MHz, 5,25-Zoll-Floppy mit 1,2 MByte, Arithmetik-Koprozessor, 51-cm-Farbbildschirm und PENS-Tastatur. Der Fileserver hat 4 bis 8 MByte RAM, 16 oder 25 MHz Taktfrequenz, 70- bis 600 MByte-Festplatte, 5,25-Zoll-Floppy, einen 31-cm-s/w-Bildschirm und eine PC-Tastatur. Die Ausgabe kann wiederum auf Laserdrucker oder Laserbelichter (z.B. Monotype PRISMA) erfolgen. Dafür können mit der Ausgabesteuerung verschiedene Formate wie PostScript, LaserSlave, ACE, Compugraphic-Slave und die Linotype-Systemsprache erzeugt werden.

Der VEB Plastunion Triptis stellte am Stand von Pneumant das Softwarepaket **Pneumant-Modul** zur rechnergestützten Konstruktion von Werkzeugen für die Verarbeitung von Plasten und Elasten, zur Simulation des Formfüllvorganges für das Spritzgießen thermoplastischer Formteile und zum Erkennen von Bindenähten und Lufteinschlüssen vor. Das Paket, das sich aus mehreren Modulen zusammensetzt, ist auf dem EC 1834 und dazu kompatiblen PCs einsetzbar und kann damit allen Betrieben, die in ihrer Fertigung Plastteile einsetzen, genutzt werden.

Zur Bearbeitung der betrieblichen Arbeitsplanstammkarten auf dem PC (8- oder 16-Bit-Technik) und für die Materialwirtschaft stellte der VEB KOSORA Dresden die Programmsysteme **STERV** und **MATPRO** vor. Damit sind die technologische Planung der Produktion und die Planung und Überwachung der Materialbestände am Arbeitsplatz des jeweiligen Mitarbeiters möglich. Das Projekt STERV bietet dabei das Archivieren, Sortieren und Ausgeben spezieller Listen und das Projekt MATPRO den Aufbau, die Korrektur und den Druck der Materialstammdaten, der Zu- bzw. Abgänge in der Materialstammdatei



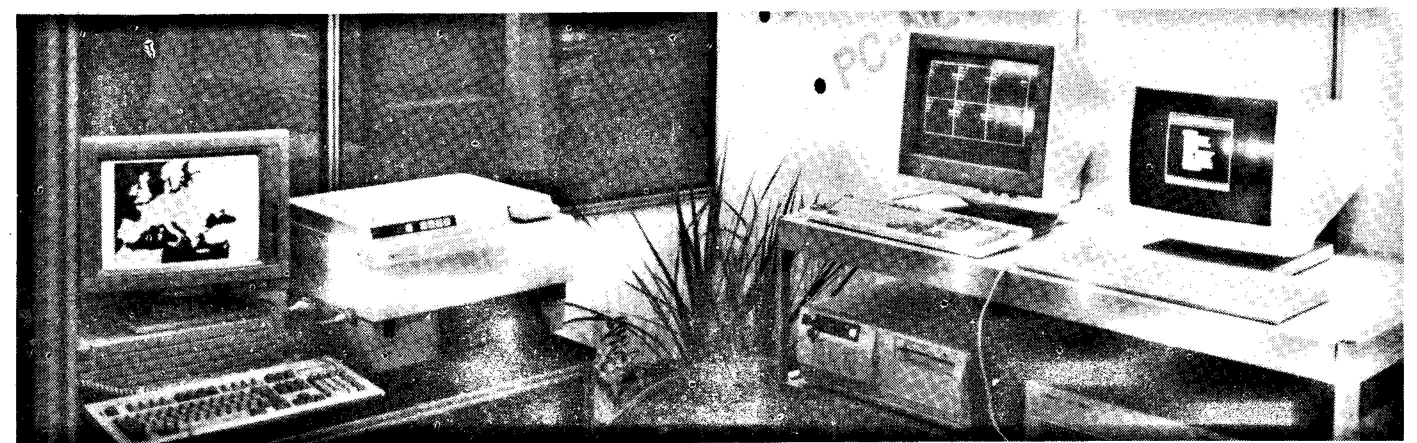
bis hin zum Druck von Inventurlisten Der Ablauf des Dialogs mit dem Computer beeinflußt die Akzeptanz eines Programmes ganz erheblich. Die Zeiten der ellenlangen Listen, die ein Großrechner ausspuckte, und der Programme, in die selbst EDV-Eingeweihte erst nach langer Einarbeitung einen Einblick erlangten, sind vorbei. Auch wenn es immer noch den einen oder anderen Programmierer geben sollte, der sein kleines Geheimnis um die Bedienung dieses Wunderwerkes nicht lüften möchte. Man darf sich dann aber immer fragen, wer hier eigentlich für wen da ist oder sein sollte. Um so lobenswerter ist es, wenn sich Softwareentwickler Gedanken darüber machen, wie nicht nur dem späteren Anwender eines Programms durch ein möglichst ähnliches Erscheinungsbild unterschiedlichster Softwareprodukte die Nutzung leichter gemacht und damit die Einarbeitungszeiten erheblich reduziert werden. Wenn dann auch den Programmierern noch Hilfsmittel in die Hand gegeben werden, kann die meist sehr aufwendig und mit viel Routinearbeit zu programmierende Dialogfähigkeit des Rechners mit geringerem Aufwand erreicht werden. Im Rahmen der Leistungsschau von Softwareprodukten der Universitäten und Hochschulen der DDR war gerade zu diesem Thema eine interessante Lösung zu sehen. Für die in unseren Betrieben nun immer häufiger anzutreffende Klasse der 16-Bit-PCs wurde von der Wilhelm-Pieck-Universität Rostock auf der Grundlage der Programmiersprache Pascal die Entwicklungsumgebung **TINA** vorgestellt (Farbbild 7). Die leichte Handhabung von TINA gestattet es insbesondere auch weniger professionellen Programmierern aus der Kenntnis ihres fachlichen Wissens heraus, den Programmablauf gestalten und auch pädagogische und psychologische Erkenntnisse bis hin zu individuellen Gewohnheiten einfließen lassen zu können.
TINA baut dann aus den Anweisungen des Anwenders die Ausschriften des Bildschirms auf. Dabei können spätere Änderungen sowohl des Bildschirmaussehens als auch des Programmablaufs leicht in das Programm integriert werden; TINA ist nach Aussage der Entwickler sehr lernfähig und äußerst tolerant, beispielsweise gegenüber Modulen in anderen Sprachen. Damit kann TINA ein wichtiges Hilfsmittel besonders bei der Erarbeitung kurzlebiger Software in hoher Qualität werden. Weitere Vorzüge sind die Anpassung an die jeweiligen Besonderheiten, nicht nur des Betriebes, und die mögliche Entwicklung von Schulungsprogrammen (sogenannten Tutoren), mit deren Hilfe die Mitarbeiter schnell und effektiv in ihre zukünftigen Aufgaben eingearbeitet werden können.

Die Simulation insbesondere technischer Prozesse kann zum einen zu einer höheren Sicherheit bei der Entscheidungsfindung führen und zum anderen die materiellen Aufwendungen für Erprobungen erheblich reduzieren. Schweißprozesse stellen dabei ein in wohl allen Bereichen der Gesellschaft angewendetes Verfahren dar. Demzufolge kommt der Simulation gerade solcher Prozesse eine große volkswirtschaftliche Bedeutung zu. Die von der Wilhelm-Pieck-Universität Rostock entwikkelte Software zur Beurteilung von Qualität und Wirtschaftlichkeit von Schweißverbindungen durch **Schweiß-ZTU-Schaubilder** beschreibt dabei sowohl das Gefüge als auch die Eigenschaftsveränderungen von Schweißgut und Grundwerkstoff einer Verbindung nach unterschiedlichen Schweißwärmezyklen. Die bisher nur als Tafelwerke existierenden Daten sind zusammen mit den Ergebnissen langjähriger Forschungsarbeit auf der Basis eines mathematischen Modells zu einem Softwareprodukt entwickelt worden, das vielfältigen Ansprüchen genügt. Die Beurteilung der Schweißbarkeit von Werkstoffen kann bereits im Stadium der Stahlentwicklung beurteilt werden, Schweißtechnologien lassen sich besser begründen, und die Eigenschaften der Schweißverbindung können vorab eingeschätzt werden.

Farbbild 8 zeigt ein typisches Diagramm, wie es unter anderem für die Abkühlzeiten, die mechanischen Eigenschaften, das Umwandlungsverhalten, die notwendige Wärmezufuhr oder für die erforderlichen Schweißdaten vom Programm berechnet werden kann.

Die Wilhelm-Pieck-Universität entwickelte ebenfalls ein modulares **Entwicklungssystem für die rechnergestützte Entscheidungsunterstützung**, deren Module über eine in modernen CIM-Lösungen eingesetzte Datenschnittstelle miteinander kommunizieren. Das vielfältig einsetzbare Werkzeug ermöglicht die effektive Entwicklung von Software für viele betriebliche Einsatzfälle und setzt sich aus einem netzfähigen Datenbankkern (DES), der Tabellenkalkulation (DES-CALC), der Datenbankanfragesprache ANNA, dem Wissensverarbeitungssystem und einem Text- und Programmeditor zusammen. Ein speziell angepaßter Prolog-Interpreter wurde in das System integriert; das Modulkonzept und die nutzeroffene Datenschnittstelle gestatten die Erweiterung des Systems mit eigenen Modulen. Die bereits vorhandenen Beispielanwendungen können an spezielle Erfordernisse angepaßt werden. Dies betrifft insbesondere eine Lösung zur Produktionsplanoptimierung (Farbbild 9), ein Literaturerfassungs- und -recherchesystem, ein Dokumentenverwaltungssystem (ebenfalls mit der Möglichkeit der Recherche), ein betriebliches Mail-System, ein Diagnosesystem für die Stomatologie und die CAD-Zeichnungsverwaltung.

Am Gemeinschaftsstand des Saarlandes wurden unter anderem Produkte von Hohe Electronics und der Karl-Marx-Universität Leipzig ausgestellt. Für die Klasse der XT/AT-PCs mit dem Betriebssystem MS-DOS und einem Arithmetikprozessor entwickelte die Karl-Marx-Universität Leipzig das wissensbasierte Rahmenexpertensystem **CAD/DSS** für die Vorbereitung von Innovationsentscheidungen sowohl in sozialistischen Industriebetrieben als auch in kapitalistischen Firmen (Farbbild 10). Das System ermöglicht insbesondere die Optimierung der Produktions- und Absatzstruktur unter Berücksichtigung der betriebsspezifischen Ziele wie maximaler Gewinnzuwachs, minimale Kosten oder maximale Rentabilitätsverbesserung unter den Bedingungen der Einhaltung vorgegebener Investitionsmittel, der Erreichung einer bestimmten Innovationsrate oder der bestmöglichen Auftragsrealisierung. Damit werden Entscheidungen der Planung der betrieblichen Entwicklung schneller und in höherer Qualität möglich; auch die so bitter notwendige schnelle Reaktion auf Veränderungen der Absatz- oder Produktionssituation kann durch die Nutzung dieses Hilfsmittels erreicht werden. Der Einsatz der Tastatur TheBoard wird empfohlen.
Wird die Tastatur TheBoard eingesetzt, stellt sich die Frage, wie die vielen Funktionen auf den einzelnen Tasten definiert werden. Dazu dient das Programm **LC Tool**, mit dem sowohl die Piktogramme, die dann auf der jeweiligen Taste erscheinen, als auch die sich dahinter verbergenden maximal 126 Zeichen (einschließlich von Steuerzeichen wie Carriage Return)



Fotos: Hemke, Hill, Schmidt, Werkfoto

in den vier möglichen Ebenen (Normal, Shift, Alt und Control) editiert werden können. Das Farbbild 11 zeigt die Programmierung einer Funktionstaste mit einem Piktogramm und die gleichzeitige Darstellung dieses Zeichens auf der LCD-Taste sowie die dazugehörige Zeichen- bzw. Befehlskette. Eine ausführliche Beschreibung der Tastatur TheBoard finden Sie im Heft 7/1989 auf der Seite 217; die Karl-Marx-Universität Leipzig, Sektion Informatik bietet im 1. Quartal 1990 einen Lehrgang für zukünftige Nutzer dieser Tastatur an. Ebenfalls an diesem Stand wurde die **Kopplung** des **EC 1834** mit der von Hohe Electronics entwikkelten Tastatur **TheBoard** über einen neuartigen, von Robotron entwickelten Interfacestecker gezeigt. Der EC 1834 war mit dem Farbbildschirm K 7233 des VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt zu sehen (Farbbild 12).

Daß auch das Kombinat Robotron ständig bemüht ist, mit einem verbesserten Design seine Produkte ansprechender zu machen und den speziellen Kundenwünschen anzupassen, zeigt die Vorstellung des neuen Nadeldruckers K 6319 und die Ankündigung des K 6325. Der **K 6319** (Farbbild 13) ist eine Variante innerhalb der K 6310-Reihe, die mit einem veränderten Gehäuse, durchsichtigem Deckel und speziellem Interfaceanschluß auf Wunsch eines Handelspartners zum Anschluß an Heimcomputer entwickelt wurde.

Bei der Entwicklung von Tischrechnern, Taschenrechnern und Pocketcomputern kann die japanische Firma Sharp bereits auf reichhaltige Erfahrungen verweisen. Neben dem bereits im Vorjahr ausgestellten Pocketcomputer PC-1600 stellte Sharp zur diesjährigen LHM den erst Mitte des Jahres auf den Markt gebrachten Electronic Organizer **IQ-7000** aus (Farbbild 14). Diese „persönliche Datenbank im Brieftaschenformat" kann über eine RS 232C-Schnittstelle mit einem PC kommunizieren (im Bild S. 382 mit dem Sharp PC-4500). Dafür bietet Sharp das Kommunikationspaket Organizer-Link (IQ-791A) an. Es besteht aus zwei Programmdisketten (3,5" und 5,25"), dem Pegelkonverter (RS 232C) und der Dokumentation. Damit ist es möglich, den IQ-7000 über einen IBM-kompatiblen PC mit Daten zu füttern oder umgekehrt Daten aus dem IQ-7000 in den PC einzulesen. Das können zum Beispiel das Adreß-/Telefonverzeichnis, Notizen, die Zeitplanung mit Alarmfunktionen oder ein langfristiger Terminplaner mit Monats-, Wochen- und Tagesübersichten sein. Dafür stehen insgesamt 32 KByte RAM zur Verfügung. Die Datenübertragung erfolgt menügeführt vom PC aus; die Daten können mit Wordstar, Sidekick oder Lotus 1-2-3 weiterverarbeitet werden. Wem das nicht reicht, der kann sich noch verschiedene IC-Karten (die EPROMs oder RAMs enthalten) zulegen: eine für Projektorganisation mit Termin- und Kontenüberwachung, eine für englische Synonyme (etwa 500000 Wörter) und als Rechtschreibhilfe (etwa 87000 Wörter), eine für einfache Übersetzungen in 8 Sprachen (rund 780 Wörter und 460 Redewendungen) sowie eine mit 32 oder 64 KByte RAM (siehe Bild 14). Die Daten des IQ-7000M werden mit drei Primärzellen (CR 2032) gepuffert; bei Batteriewechsel bleiben die Daten erhalten. Der RAM-Modul enthält eine CR 2016, die 2 Jahre die Daten puffert und wechselbar ist.

Die LC-Anzeige hat 8 Zeilen zu je 16 Zeichen und ist grafikfähig. Zur diesjährigen SYSTEMS '89 im Oktober in München stellte Sharp bereits die Weiterentwicklung IQ-7100 vor. Hier kann für die Anzeige auf dem Display eine von 7 Sprachen vorgewählt werden.

Der VEB Organisation und Rechentechnik Karl-Marx-Stadt – ein Softwarehaus, das sich unter anderem auch auf die Installation von Computern spezialisiert – zeigte eine interessante **LAN-Lösung unter Einbeziehung des EC 1834** als Nutzerstation. Die Hardwarevoraussetzungen dafür bildeten ein mit 16 MHz getakteter 286er AT als File- und Druckserver und ein originales Ethernet (siehe Bild oben). Als Netzsoftware wurde die SFT Advanced NetWare 2.15 von Novell verwendet. In ein Netz mit dieser Konfiguration lassen sich 30 bis 50 EC 1834 einbinden, die gleichberechtigt solche Dienste, wie Fileserving, Spooling und Printserving, Verwaltung der Zugriffsberechtigung, Accouting und andere in Anspruch nehmen können. Wichtigste Vorteile einer solchen Lösung sind:

– alle Nutzer haben die gleiche Datenbasis und
– teure Peripheriegeräte müssen im Netz nur einmal vorhanden sein.

Großes Interesse fand auch die von diesem Betrieb vorgestellte Lösung zum Datenschutz für PCs mit Harddisk. Die angebotene Software dazu nennt sich **SAVE**, arbeitet unter MS-DOS und DCP und schützt die Festplatte durch die Vergabe eines Paßwortes vor unberechtigten Zugriffen.

Zur Frühjahrsmesse '89 war erstmalig der niederländische Computerhersteller Tulip vertreten. Damals mit einem eigenen Stand, war er zur Herbstmesse bei der Handelsfirma Transcommerz aus Berlin (West) – die daneben vorrangig Kopiertechnik anbot – zu finden. Seit Juni dieses Jahres ist Transcommerz offiziell Distributor von Tulip computers. Tulip stellt PCs aller Leistungsklassen, Netze und Multiusersysteme her. Besonderes Gewicht legt die Firma auf die Qualitätskontrolle der Geräte und einen hohen technischen Standard. Alles Gründe, die Tulip in wenigen Jahren an die dritte Stelle der westeuropäischen Computerhersteller gebracht haben. Das herausragende Produkt von Tulip war das Ethernet-kompatible **Isolan**, das mit seiner Fülle von Komponenten an jeden Anwendungsfall angepaßt werden kann. Herzstück war als Fileserver der Tulip at 386sx (im Farbbild 15 links), den wir auf der 4. Umschlagseite vorstellen. Mit dem Server im Netz verbunden waren der pc compact2, der at compact2 (beides Weiterentwicklungen der zur LFM gezeigten Modelle pc compact und at compact) und der Tulip isolan (im Bild 15 rechts) – eine festplattenlose Variante des at mit Isolan-Netzkarte und BIOS-Boot-ROM; das bedeutet, daß beispielsweise beim Einschalten des Rechners das Betriebssystem nicht von Diskette oder Festplatte, sondern aus dem nichtflüchtigen Speicher (ROM) geladen wird.

Ein Ethernet-kompatibles Netz besteht in seiner Grundausstattung aus dem LAN-Kabel (Koaxialkabel oder Lichtleiter), der Ankoppelbaugruppe an das Kabel (Transceiver), der Stichleitung vom Rechner zum Transceiver und der Netzkarte im PC. Auf diese Weise können mit dem Isolan bis zu 100 Rechner über 500-m-Segmente miteinander gekoppelt werden. Das Isolan bietet aber wesentlich mehr. Weil die Isolan-Karten für den XT und den AT einen Transceiver enthalten, kann eine Billigvariante (Cheapernet) – bei der der separate Transceiver wegfällt und die PCs direkt über BNC-Buchse und Koax-Kabel (Thin Ethernet Cable) verbunden werden – bis zu 30 Rechner über 185-m-Segmente miteinander verbinden. Die weiteren Baugruppen sollen hier kurz erwähnt werden:

– Fan-Out Unit (sternförmiges Vernetzen von bis zu 8 PCs und Ankopplung an das Isolan)
– Multiplexer (asynchroner serieller Anschluß für mehrere Terminals, Drucker oder Modems)
– MCA Controller Card (16-Bit-Erweiterungskarte für das IBM PS/2; ebenfalls mit On-board-Transceiver für das Cheapernet)
– Repeater (Verbinden von zwei LAN-Segmenten über größere Entfernungen)
– Multi-Port Repeater (Verbinden von mehreren LAN-Segmenten; mit Interfaces für alle Kabeltypen)
– Primary Bridge und Managed Bridge (Verbinden von zwei LANs)
– Remote Bridge (Schnittstellen: X.25, V.24, 2 × 64 kBit/s)

Für Transceiver, Fan-Out Unit, Repeater und Multi-Port Repeater sind Versionen für Lichtleiter verfügbar.

Zu der von Tulip angebotenen Software gehören die ELS II NetWare für bis zu acht Nutzer und die Advanced NetWare für 100 LAN-Teilnehmer. Die SFT NetWare enthält Schutzmechanismen beispielsweise gegen Ausfälle der Hardware, und die Multi-Protocol Support Software (MPS) ermöglicht die Kommunikation mit anderen Systemen, sowohl mit anderen LANs als auch mit Mini- und Großrechnern – und das sogar simultan.

Eine Besonderheit für das LAN stellt auch der System Control Manager (SCM) dar, ein anwendungsspezifischer Schaltkreis, der auf der 4. Umschlagseite näher erläutert wird. Beim Abschalten des Servers wird der Bediener nicht nur nach dem Paßwort gefragt, sondern er kann auch noch die Zeit für die Ausschaltverzögerung zwischen 0 und 255 Minuten wählen. Beim Start der Ausschaltverzögerung werden sämtliche Teilnehmer des Isolans automatisch mit einer Nachricht darüber informiert, wann der Server abschaltet. Dadurch besteht für alle Nutzer die Möglichkeit, die Arbeit mit den Massenspeichern des Servers rechtzeitig zu beenden.

Im 40. Jahr der Beteiligung Großbritaniens an der Leipziger Messe war die Ausstellung British Design ein Novum. Mit Unterstützung des Central Office of Information wurden auf 150 m² Ausstellungsfläche Spitzenleistungen von 26 Industriedesignern vorgestellt. In Großbritannien hat sich das Design in den frühen 80er Jahren zu einem eigenständigen Wirtschaftszweig entwickelt. Heute gibt es rund 2000 Design-Beratungsbüros in Großbritannien. Für die Leser der MP interessante Produkte waren der im Farbbild 16 dargestellte Handcomputer **Psion Organiser II** (links) mit einer 32stelligen LC-Anzeige und der **Psion Printer II** (rechts).

Der Organiser II ist wie der IQ-7000 von Sharp praktisch ein sehr komfortabler Taschenrechner. Statt der IC-Karten gibt es hier sogenannte Memory Packs und Program Packs (unten rechts im Bild), von denen aber im Unterschied zum IQ-7000 zwei gleichzeitig in den Organiser II gesteckt werden können. Die Memory Packs können RAM-Größen von 8 bis 128 KByte enthalten. Das Modell LZ64 (mit 64 KByte internem RAM) kann demnach mit maximal 320 KByte On-board-Speicher ausgerüstet werden. Außerdem hat der interne ROM eine Größe von maximal 64 KByte. Sieben verschiedene Program Packs sind unter anderem für finanzielle Berechnungen, für englische Rechtschreibung (mit über 24 000 Wörtern), für einfache Übersetzungen in 5 Sprachen (mit über 700 Wörtern und Redewendungen) und für mathematische Berechnungen verfügbar. Der Organiser II kann natürlich ebenfalls über eine RS 232-Schnittstelle mit PCs oder auch direkt mit Druckern oder Modems gekoppelt werden. Aber Psion bietet auch für unterwegs eine Druckmöglichkeit mit dem Printer II. Er kann 80 Zeichen pro Zeile im Normalmodus drucken. Im Grafikmodus sind 20, 40, 60 und 80 Spalten mit einer Auflösung von 256 Punkten pro Linie möglich. Die Geschwindigkeit ist 1 Zeile/s. Ein Wechselspannungsadapter ermöglicht den Netzbetrieb; die internen 9-V-Batterien erlauben nur einen Betrieb von 2 bis 6 Monaten. Programmiert werden kann der Organiser II mit der Programmiersprache OPL – einem Basic-Derivat. Die Software Organiser II Developer ermöglicht für OPL das Entwickeln, Testen und Beseitigen von Fehlern auf IBM-kompatiblen PCs.

H. Hemke, H.-J. Hill

Atari-Messe 1989

Wie breit das aktuelle Angebotsspektrum rund um die Atari-Computer inzwischen ist, wurde vom 25. bis 27. August dieses Jahres wiederum auf der Atari-Messe in Düsseldorf – dem umfassendsten Angebotsforum der Firma – deutlich. Über 150 Aussteller zeigten vom einfachen Videospiel bis zum professionellen Einsatz im technisch-wissenschaftlichen Bereich die neuesten Softwareentwicklungen und Peripheriegeräte. Natürlich waren alle bekannten Atari-Computer im Einsatz zu sehen; der Hersteller nutzte diese Gelegenheit jedoch auch zur Präsentation seiner neuesten Produkte.

Da wäre zunächst die vom Geschäftsführer Alwin Stumpf als spektakulärste Neuigkeit angesehene Vorstellung des **Atari TT** als Ergänzung zur ST-Reihe (Bild 1). Der TT basiert auf Motorolas 68030-Mikroprozessor, der mit 16 MHz getaktet wird. Zur ST-Reihe ist der Rechner voll kompatibel und bietet alle Möglichkeiten zum Anschluß der Atari-Peripherie oder entsprechender Fremdperipherie. Besonders unterstützt wird die Verwendung im Musikbereich durch spezielle Hardwarekomponenten: Yamaha Soundchip YM-2149, interne Lautsprecher, aktive Stereo-Audio-Buchse, Midi-Anschlüsse. Weitere Merkmale: 2 MByte RAM als Standard, ausbaufähig bis zu 8 MByte, Farbpalette mit 4096 Farben bei einer Bildschirmauflösung von 320 × 200 bis zu 640 × 400 Punkten, hochauflösender Monochrommodus mit 1 280 × 960 Punkten, 8-Bit-PCM-Stereoklanggenerator, 1 × VMEbus-Erweiterungsslot. Als Betriebssystem wird das zum ST/TOS kompatible TOS 030 mitgeliefert. Als eine Version des TT wird zum ersten Quartal 1990 die schon lange angekündigte Unix-Maschine, der Atari TT/X erwartet. Laut Stumpf war die Verzögerung darauf zurückzuführen, daß die 68020-Linie nicht weiterverfolgt wurde. Der TT/X wird sich vom TT vor allem durch das Towergehäuse, mehrere VMEbus-Steckplätze und mindestens 4 MByte RAM unterscheiden.



①

Die zweite Weltpremiere gab es mit der Vorstellung des **Atari STE** (Bild 2), ausgestattet mit Verbesserungen gegenüber seinem Bruder 1040. Beispielsweise bietet der STE hardwaregestütztes horizontales und vertikales Scrolling. Die Farbpalette umfaßt 512 aus 4096 Farben; die Bildschirmauflösung beträgt bis zu 640 × 400 Punkte. Diese Bildqualität, Lichtpistole, neue Joysticks und der Stereosound in CD-Klangqualität deuten auf den anvisierten Markt der Computer-Aktionsspiele hin. So können sich vier Spieler gleichzeitig an einem STE betätigen, und eine Erweiterung auf sechs Spieler ist möglich.

Dennoch möchte die Firma den Markt der kompatiblen PCs nicht vernachlässigen: Die Prototypen einiger neuer Modelle sollen bereits im Test sein.

Zur CeBIT '89 erregte Atari mit der Vorstellung „des ersten kompatiblen Personal Computers für die Westentasche" Aufsehen. Der damals noch als PC Folio bezeichnete Prototyp wurde zur Atari-Messe nunmehr ausgereift als **Portfolio** präsentiert (Bild 3). Er wird als absolut vollwertiger PC bezeichnet, der mit dem Betriebssystem MS-DOS 2.11 arbeitet. Bei der Grundfläche von 20 × 10 cm² und 2,7 cm Höhe kann man sich natürlich vorstellen, daß für ein herkömmliches Floppylaufwerk kein Platz bleibt. Die Firma hat sich daher für Speicherkarten im Scheckkartenformat entschieden, die Kapazitäten von 32, 64 oder 128 KByte aufweisen und vom Rechner wie Disketten behandelt werden. Eine in den Karten enthaltene Batterie hält die Daten etwa ein Jahr lang. Daneben wird als RAM-Disk (Simulation der Festplatte) ein Teil des 128-KByte-RAMs (aufrüstbar auf 640 KByte) genutzt. Der 256 KByte große ROM enthält bereits mehrere Anwendungsprogramme wie Lotus 1-2-3-Tabellenkalkulation, Textverarbeitung, Adreßverwaltung oder ein Kommunikationsprogramm. Damit ist die Kommunikation mit anderen PCs oder der Peripherie möglich. Hardwaremäßig gibt es dafür eine kombinierte serielle/parallele Schnittstelle. Als Zentraleinheit wird die stromsparende CMOS-Version Intel 80C88 verwendet, so daß sich der Rechner, laut Hersteller, sechs bis acht Wochen mit den drei 1,5-Volt-Batterien betreiben läßt (für Netzbetrieb gibt es ein separates Netzteil). Das LC-Display in Supertwisttechnik bietet eine Auflösung von 240 × 64 Punkten und die Darstellung von 320 Zeichen.

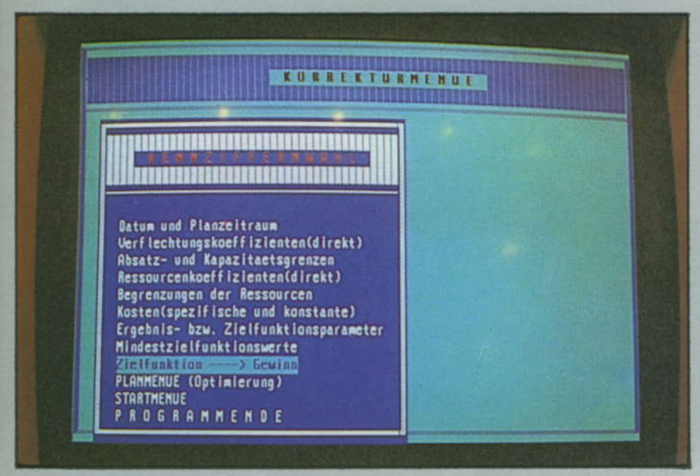
Als Preis für diese 495 Gramm Hochtechnologie im Taschenformat werden 800 DM angegeben. *MP-We*



②



③



9 **Entwickeln rechnergestützter Systeme zur Entscheidungsfindung**



10 **Analyse und Optimierung betriebswirtschaftlicher Daten**



11 **TheBoard wird mit LC Tool individuell programmiert**



12 **EC 1834 mit Farbbildschirm K 7233 und TheBoard**



13 **Matrixdrucker K 6319 von Robotron**



14 **Electronic Organizer IQ-7000 von Sharp**



15 **Am Stand von Transcommerz: Lokales Netz Tulip isolan**



16 **Handcomputer Organiser II und Printer II von Psion**

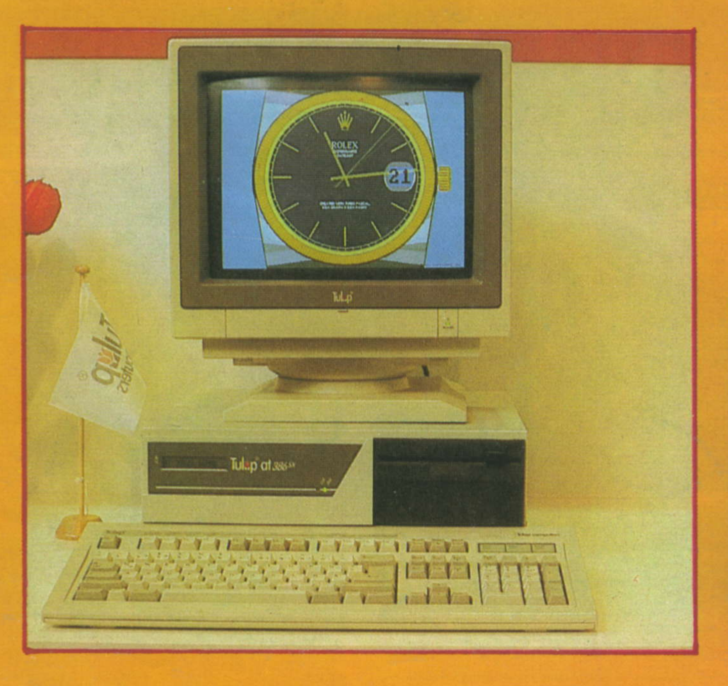
Lesen Sie hierzu unseren Bericht von der LHM '89.

ISSN 0232-2892
Mikroprozessortechnik, Berlin 3 (1989) 12
S. 353–384

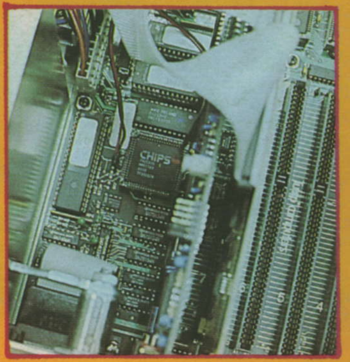
vorgestellt

Tulip at 386sx

Als Hersteller von solide gebauten Personalcomputern (PCs) hat die noch relativ junge niederländische Firma Tulip computers inzwischen einen guten Namen. In unseren Berichten von der Leipziger Frühjahrsmesse, von der CeBIT'89 sowie von der diesjährigen Leipziger Herbstmesse (siehe Innenteil dieses Heftes) sind wir mehrfach auf die mittlerweile recht umfangreiche Produktpalette eingegangen. Der hier vorgestellte at 386sx ist als Einstiegsmodell in die Welt der 32-Bit-PCs anzusehen. Seit dem Erscheinen des Mikroprozessors 80386SX von Intel – wir berichteten in MP 9/88 darüber – wird er von vielen PC-Produzenten als Basis für eine Leistungsklasse zwischen den 16-Bit-PCs mit dem Prozessor 80286 (ATs) und PCs mit dem „echten" 32-Bit-Prozessor 80386 genutzt. Bekanntlich wurde beim SX die Anzahl der externen Datenleitungen von 32 auf 16 reduziert, was die Fertigung des Prozessors vereinfacht – sprich verbilligt – und zudem die Nutzung der herkömmlichen peripheren AT-Baugruppen zuläßt. Da der Prozessor intern jedoch mit 32 Bit arbeitet, kann andererseits die gesamte, und vor allem künftige, Software für 80386-PCs verwendet werden. Ob SX-PCs für ATs auf Dauer eine echte Konkurrenz sein werden, ist schwer abzuschätzen, da der Prozessor 80286 von verschiedenen Herstellern ständig verbessert wird und mittlerweile mit bis zu 20 MHz Taktfrequenz lieferbar sein soll. Der 80386SX wird lediglich in 16-MHz-Ausführung gefertigt. Tulip sieht den Einsatzschwerpunkt des at 386sx bei umfangreichen Datenanwendungen als Server im Netz, wofür 32-Bit-PCs gute Voraussetzungen bieten. Dank des großen Arbeitsspeichers – 1 MByte vorhanden, auf der Platine bis auf 5 MByte erweiterbar – eignet er sich auch für Anwendungen, die viel Speicher benötigen, beispielsweise CAD- und DTP-Anwendungen oder Software unter dem Betriebssystem OS/2. Dazu trägt ebenfalls die serienmäßig vorhandene Zusatzspeicher-Verwaltung EMS 4.0, die auch als LIM-Standard bekannt ist, bei. Wird die EMS-Funktion nicht benötigt, kann der RAM-Bereich zwischen 640 KByte und 1 MByte genutzt werden, um das BIOS oder EGA- bzw. VGA-ROMs zu beschleunigen (Shadow-RAM).



Gegenüber dem kleineren Bruder, dem SX compact2, bietet der at 386sx zum einen mehr freie Steckplätze für Erweiterungsleiterkarten – sechs 16-Bit-Slots (AT-kompatibel) und zwei 8-Bit-Slots (XT-kompatibel) für volle Länge –, zum anderen vor allem ein bisher einzigartiges Datenschutzsystem. Kernstück dieses Sicherheitsmechanismus, System Control Manager (SCM) genannt, ist ein von Tulip entwickelter anwendungsspezifischer Schaltkreis (im rechten Bild auf der linken Seite zu sehen), mit dessen Hilfe laut Tulip ein 100prozentiger Systemschutz gewährleistet wird. Für den Nutzer stellt sich der SCM als 16stelliges LC-Display an der Frontseite des Computers dar, über das die Kommunikation mit dem Sicherheitssystem erfolgt. Das heißt, das Sicherheitssystem gibt über das Display Meldungen über den Betriebszustand des Computers oder Hinweise für auszuführende Handlungen des Benutzers aus. So tritt beispielsweise das LC-Display mit der Ausschrift STANDBY in Aktion, wenn der Netzschalter betätigt wurde, die Hauptplatine des Rechners aber noch ausgeschaltet ist. Mit einer (vom Benutzer wählbaren) Tastenkombination kann zunächst der SCM aktiviert werden; die Ausschrift lautet dann: ENTER FUNCTION. Mit der Taste F1 kann nun die Einschaltfunktion gewählt werden; Ausschrift: POWER UP<–. Nach Drücken der Entertaste erscheint jetzt: ENTER PASSWORD. Bis hierhin ist der Bildschirm des Computers immer noch dunkel. Erst nach der Eingabe des Paßwortes schaltet der SCM den Rechner ein. Ähnlich gestaltet sich die



Ausschaltprozedur. Hinzu kommen aber noch die Möglichkeit der Wahl der Ausschaltverzögerung von bis zu 255 Minuten (Näheres hierzu in unserem Bericht von der LHM'89) bzw. des Einbaus des Ausschaltbefehls SCMDOWN in ein Batchfile. Letzteres ermöglicht einen pünktlichen Feierabend, da dieses Batchfile das Anlegen einer Sicherheitskopie (beispielsweise der gesamten Festplatte auf Magnetband) und hinterher das Ausschalten des Rechners beinhalten kann. Insgesamt bietet das SCM-System also weit mehr, als dies von herkömmlichen Zugangssperren mittels Paßwort bekannt ist. So läßt sich das System auch nicht durch Ab- und Wiedereinschalten des Stromnetzes oder durch Booten von der Diskettenstation aus überlisten. Was jedoch nicht passieren darf: Der Nutzer hat sein Paßwort vergessen. Da hilft nur noch der Servicemechaniker, der den Rechner wieder in den Grundzustand bringt, so daß der Kunde ein neues Paßwort eingeben kann.

Technische Daten

Prozessor	Intel 80386sx (8 oder 16 MHz, mittels Software umschaltbar); ca. 0,7 Wartezyklen
RAM	1 MByte auf Hauptplatine; mittels SIMMs bis zu 5 MByte auf Hauptplatine erweiterbar
Floppy	1 Laufwerk, wahlweise 5,25 Zoll (360 KByte oder 1,2 MByte) oder 3,5 Zoll (720 KByte oder 1,44 MByte)
Festplatte	1 Laufwerk 3,5 Zoll, 40 MByte, < 25 ms, ST-506-Interface
Schnittstellen	1 × parallel Centronics 2 × seriell RS 232C

Fotos: Hemke, Weiß

MP